AMSAA

TECHNICAL REPORT NO. TR-694

DEVELOPMENT OF THE EXPECTED ACCIDENT AGGREGATION MODEL

FEBRUARY 2002

U.S. ARMY MATERIEL SYSTEMS ANALYSIS ACTIVITY
ABERDEEN PROVING GROUND, MARYLAND 21005-5071

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY (LEAVE BLANK)	2. REPORT DATE February 2002	3. REPORT TYPE AND DATES COVERED Technical Report	
4. TITLE AND SUBTITLE Development of the Expected Accident Aggregation Model			5. FUNDING NUMBERS
6. AUTHOR(S) Dr. Michael J. Cushing			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Director U.S. Army Materiel Systems Analysis Activity 392 Hopkins Road Aberdeen Proving Ground, MD 21005-5071			8. PERFORMING ORGANIZATION REPORT NUMBER TR-694
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Director U.S. Army Materiel Systems Analysis Activity 392 Hopkins Road Aberdeen Proving Ground, MD 21005-5071			10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE A
13. ABSTRACT (Maximum 200 words) This report documents the development of the Expected Accident Aggregation Model. This model was developed in order to aggregate helicopter fleet System Safety Risk Assessments (SSRA). Expected-accident quantities and probabilities are generated for various categories of SSRAs, in aggregate, and as a function of time. This report includes a basic set of electronic templates written in *Mathematica*, a leading commercial software package, for applying the new model to helicopter fleets.			
14. SUBJECT TERMS			15. NUMBER OF PAGES
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT SAME AS REPORT

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

THIS PAGE INTENTIONALLY LEFT BLANK.

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK.

Chapter 1

Introduction

To support the Aviation Safety Integrated Process Team, the Aviation and Missile Research, Development and Engineering Center asked AMSAA to develop a model that aggregates the risks due to the collection of open SSRAs applicable to a helicopter fleet at a point in time. SSRAs are prepared when an unexpected materiel risk is discovered. SSRAs do not address risks due to the environment, personnel, doctrine or training. AMSAA developed two models to aggregate the risks of individual SSRAs. These models can be used to calculate accident probabilities due to SSRA causes as SSRAs are identified and resolved through time. This will allow decision makers to consider the cumulative risk to the fleet due to all open SSRAs when deciding which risk-mitigation option to select for a new SSRA.

Chapter 2 provides background on the need for a new model and describes the development of two models for aggregating SSRAs. The first, more complex model is termed the Component-Socket Hazard model. This model can provide helicopter-level accident predictions as a function of time at the cost of significant computational complexity. Unfortunately, it requires component aging models and component-age data, information not currently available for all SSRAs. It is possible to apply this model in the future when the requisite data becomes available.

The simpler model, termed the Expected Accident Aggregation (EAA) model, can provide fleet-level predictions. Since it was possible to apply this model with modest enhancements to current SSRA data, it is the primary focus of this report. In fact, AMSAA Technical Report 695 applies this model to the SSRAs for the Apache helicopter fleet. Application of the EAA model to a notional helicopter fleet is addressed in Chapters 3 and 4 of this report. Chapter 3 contains the input and examination of the SSRA input data whereas Chapter 4 covers the aggregation and analysis of said data. The report is concluded in Chapter 5.

The electronic form of each chapter and appendix of this report is a *Mathematica* 4 notebook. All of the methodology, computations and graphics in this report are *Mathematica* executables. Thus the technical content of this report is "live" in the sense that it can be re-executed as desired by readers working with the electronic version (provided they have a copy of *Mathematica* 4). Appendix A contains the *Mathematica* functions for the EAA model. Chapters 3 and 4 may be used as electronic templates for the input and analysis of SSRA data. The results were generated and inserted by *Mathematica*. Please refer to *The Mathematica Book* [Wolfram 1999] for information on this software. Additional information, including a free reader, is available at *http://www.wolfram.com/*.

In order to execute the electronic version of this report, it must first be installed as follows: Create a directory `AviationSafety` in your *Mathematica* `ExtraPackages` directory; create a directory `ExpAccAggModel` in the `AviationSafety` directory, then create a directory `MethodologyTR` in the `ExpAccAggModel` directory; copy the entire technical report into the `MethodologyTR` directory; the two files `Appendix A.nb` and `Appendix A.m` should be copied into the `AviationSafety` directory and renamed `ExpectedAccidentAggregationModel.nb` and `ExpectedAccident-AggregationModel.m`, respectively.

Chapter 2

Model Development

Introduction

In support of the Aviation Safety Integrated Process Team (IPT), the Aviation and Missile Research, Development and Engineering Center (AMRDEC) asked AMSAA to develop a model that aggregates the risks due to the collection of open SSRAs applicable to a helicopter fleet at a point in time. SSRAs are prepared when an unexpected materiel risk is discovered. SSRAs do not address risks due to the environment, personnel, doctrine or training. AMSAA developed two models to aggregate the risks of individual SSRAs. These models can be used to calculate accident probabilities due to SSRA causes as SSRAs are identified and resolved through time. This will allow decision makers to consider the cumulative risk to the fleet due to all open SSRAs when deciding which risk-mitigation option to select for a new SSRA. The two models will be presented in this chapter after the need for the new model is considered in greater detail.

Need for a New Model

When a new SSRA is presented to a senior Army leader for the selection of a risk-mitigation option and approval, the leader typically wants to know what the aggregate level of risk is due to all of the SSRAs currently open. This is problematic since no model currently exists for the aggregation of SSRAs. The AMRDEC, the preparers of SSRAs, asked AMSAA to develop a model that aggregates the individual risk from each SSRA. Some background on SSRAs is in order.

An SSRA applies to a helicopter fleet, not an individual helicopter. When a new or previously-undiscovered materiel risk to a helicopter fleet is identified, an SSRA is typically prepared by the AMRDEC. SSRAs are therefore prepared at random points in time. The risks associated with an SSRA may increase over time due to factors such as:

- delays in component replacements,
- component aging,
- reduced inspections,
- increases in fleet size and/or
- an increase in helicopter flight hours.

SSRA risks may decrease due to factors such as:

- earlier component replacements,
- increased inspections,
- decreases in fleet size and/or
- decreases in the number of flight-hours per helicopter.

An SSRA categorizes risks in terms of both severity-level and likelihood of occurrence. The categories used are depicted in the typical risk matrix below.

		Hazard Probability (Frequency)				
		Frequent	Probable	Occasional	Remote	Improbable
Severity		A	B	C	D	E
I	Catastrophic	■	■	■		
II	Critical	■	■			■
III	Marginal				■	■
IV	Negligible	■	■	■	■	■

At present, the "Hazard Probability (Frequency)" for an SSRA may be quantified in a variety of ways.

- failure probability at the time when the component is to be replaced or at the end of the helicopter life-cycle,
- reliability at the time when the component is to be replaced or at the end of the helicopter life-cycle,
- expected number of accidents for the entire fleet over the entire helicopter life-cycle.

In some cases, only a qualitative likelihood of accident occurrence is developed.

Sometimes calendar time is important, for example, to predict how many severity-level I accidents can be expected to occur in a specific helicopter fleet in 2003. Sometimes it is important to predict risk based on flight hours, for example, to predict the number of accidents one should expect in the next 100,000 flight hours. Expressing risk based on flight hours is desirable when comparing large and small fleets.

A new model that has the following characteristics is needed:

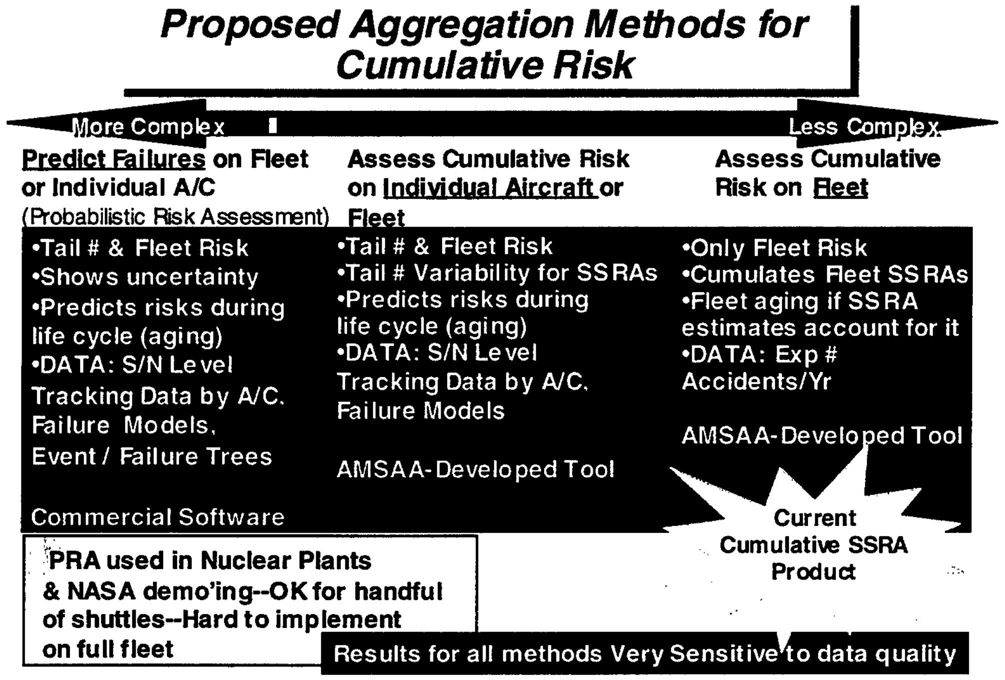
- addresses risks from multiple severity levels,
- relates risk to both calendar time and flight hours,
- accounts for SSRA risks being identified and resolved asynchronously in time,
- allows risks to increase and/or decrease over time,
- requires assumptions that are relatively few, weak and likely to be met,
- doesn't flow assumptions down to the SSRA process,
- leaves the SSRA experts free to innovate their process.

Modeling Options

Three modeling approaches for aggregating SSRA risks were identified and considered. They are depicted in the figure below. The most intensive approach (left-most column in figure) would be to incorporate each SSRA into a complex helicopter reliability model that accounts for redundancies, dependencies and aging. Such models are often developed during system development using Probabilistic Risk Assessment (PRA) techniques, but are not necessarily kept up-to-date during the rest of the life cycle. (Kumamoto and Henley's 1996 text is an excellent reference on PRA techniques.) This approach, while the most powerful, was not pursued in this study since it did not seem feasible given the available data and time.

The application of a simpler modeling approach (center-most column in figure) resulted in the development of the Component-Socket Hazard model. This model, described in the next section, was not applied to an actual helicopter because component hazard functions and tracking data were not readily available. This model is more powerful than the model actually applied (see below) in that it gives one the ability to explicitly model risks over time on specific helicopters, including the effects of aging. Perhaps it will be possible to implement this model in the not-too-distant future as component tracking improves.

The application of the simplest modeling approach (right-most column in figure) resulted in the development of the Expected Accident Aggregation model, which is addressed in a subsequent section of this chapter. This model was the easiest to apply to the existing SSRA process in that only minor enhancements of the current SSRA process were required. Application and illustration of this model with a notional helicopter fleet is the focus of the rest of this report. It was also applied to the Apache helicopter fleet in AMSAA Technical Report 695.



Proposed Aggregation Methods for Cumulative Risk

More Complex ← → Less Complex

Predict Failures on Fleet or Individual A/C (Probabilistic Risk Assessment)	**Assess Cumulative Risk on Individual Aircraft or Fleet**	**Assess Cumulative Risk on Fleet**
•Tail # & Fleet Risk •Shows uncertainty •Predicts risks during life cycle (aging) •DATA: S/N Level Tracking Data by A/C, Failure Models, Event / Failure Trees Commercial Software	•Tail # & Fleet Risk •Tail # Variability for SSRAs •Predicts risks during life cycle (aging) •DATA: S/N Level Tracking Data by A/C, Failure Models AMSAA-Developed Tool	•Only Fleet Risk •Cumulates Fleet SSRAs •Fleet aging if SSRA estimates account for it •DATA: Exp # Accidents/Yr AMSAA-Developed Tool

PRA used in Nuclear Plants & NASA demo'ing--OK for handful of shuttles--Hard to implement on full fleet

Current Cumulative SSRA Product

Results for all methods Very Sensitive to data quality

Component-Socket Hazard Model

The Component-Socket Hazard model aggregates component hazard functions and can assess the SSRA risk associated with each specific helicopter and also assess the fleet-level risk. This model can address helicopter-to-helicopter variability and will calculate the risks at any point in the life-cycle. However, the model requires component aging models and component-age data. Component-aging models and component-age data are not currently available for all of the SSRA components. When the data become available, a highly-automated version of the model, suitable for routine application to the helicopter fleets, could be prepared.

■ Theoretical Description

Component

A continuous time-to-accident probability distribution can be expressed in the form of a hazard function. As will be seen later, it is quite useful to do so. Consider a hypothetical component installed on a helicopter with a time-to-accident distribution that can be modeled with a two-parameter Weibull distribution. Should this component fail, an accident of a particular severity level will occur. This component is a replacement for the component the helicopter was originally equipped with. The original component was removed due to scheduled maintenance when it reached the specified age. The currently-installed component is believed to be weaker than the original and an SSRA was prepared. When the component reaches its replacement age, it will be removed and replaced by a component equal to the original and the SSRA will no longer apply to the helicopter.

Functions for the Weibull distribution are available in the standard add-on package *Statistics`ContinuousDistributions`* which is loaded thus:

```
Needs["Statistics`ContinuousDistributions`"]
```

The usage message for the Weibull distribution is obtained thus:

```
?WeibullDistribution

WeibullDistribution[alpha, beta] represents the Weibull
  distribution with shape parameter alpha and scale parameter beta.
```

With a shape parameter greater than one, the Weibull distribution can be used to characterize aging. Let us suppose we have a component with shape and scale parameters of 3 (a dimensionless number) and 5,000 hours, respectively. This distribution is assigned as the value of the symbol *wbldist* for convenience:

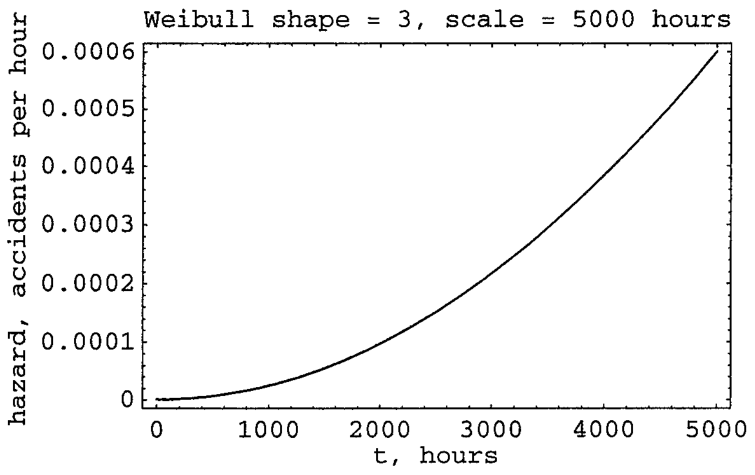
```
wbldist = WeibullDistribution[3, 5000]

WeibullDistribution[3, 5000]
```

An explicit definition for the hazard function is not available in the add-on package *Statistics`ContinuousDistributions`*. Fortunately, the hazard function can be readily obtained from the probability and cumulative density functions:

$$\text{wblhaz[t\_]} := \frac{\text{PDF[wbldist, t]}}{1 - \text{CDF[wbldist, t]}}$$

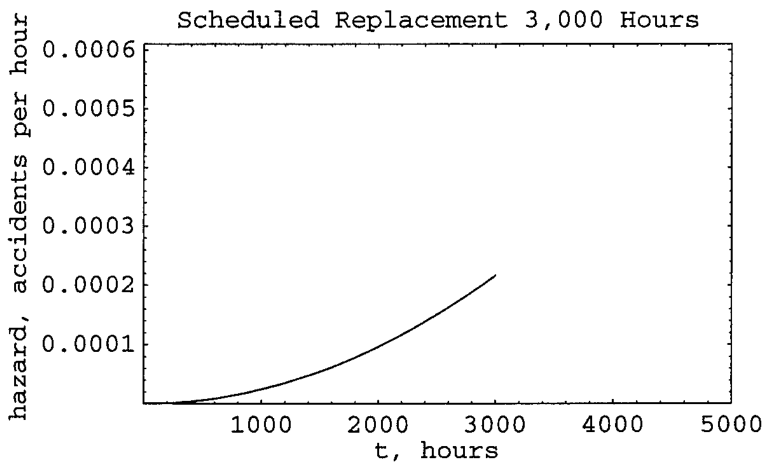
A plot of this hazard function is:

```
Plot[wblhaz[t], {t, 0, 5000}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", " hazard, accidents per hour",
    "Weibull shape = 3, scale = 5000 hours", None},
  PlotStyle → RGBColor[0, 0, 1]];
```



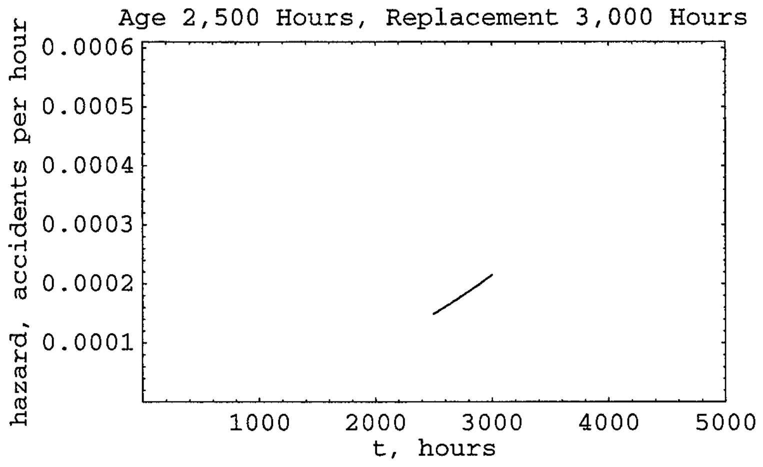
Critical components that are subject to aging are often replaced when they reach a particular age. Let's assume that our hypothetical component will be replaced when it reaches an age of 3,000 hours. We can generate a revised hazard plot thus:

```
Plot[wblhaz[t], {t, 0, 3000},
  PlotRange → {{0, 5000}, {0, .00061}}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", " hazard, accidents per hour",
    "Scheduled Replacement 3,000 Hours", None},
  PlotStyle → RGBColor[0, 1, 0]];
```



The graph above plots the hazard function from when the component is new until it is 3,000 hours old. The riskiest portion of the hazard function is now excluded. It may also be helpful to consider situations where the component is not new. This is useful when one will be periodically called upon to provide updated accident-probability calculations for a component that is already in-service. Suppose that our hypothetical component already is already 2,500 hours old. The hazard plot is then:

```
Plot[wblhaz[t], {t, 2500, 3000},
  PlotRange → {{0, 5000}, {0, 0.00061}}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", "hazard, accidents per hour",
    "Age 2,500 Hours, Replacement 3,000 Hours", None},
  PlotStyle → RGBColor[0, 1, 0]];
```



The portion of the hazard function that is now in the past is excluded from the plot above. That which remains is the hazard for a component with a current age of 2,500 hours that is to be replaced at 3,000 hours.

Of what further use is the hazard function? For starters, it can be used to calculate and plot the probability that the component will or will not survive until replacement. How might one plot and calculate the probability that an accident does not occur for this situation? The conditional reliability of a component of age t_1 surviving to t (where $t > t_1$) is $\frac{R(t)}{R(t_1)}$. The reliability can be calculated directly from the hazard function using the formula $R(t) = Exp[-\int_0^t h(u)\,du]$. Using this formula in the ratio above yields:

$$\frac{e^{-\int_0^t h[u]\,du}}{e^{-\int_0^{t1} h[u]\,du}}$$

$$e^{-\int_0^t h[u]\,du+\int_0^{t1} h[u]\,du}$$

Since t is greater than t_1, the first integral can be broken into two integrals, one over the limits 0 and t_1 and the other over the limits t_1 and t:

$$\% \,/.\, \int_0^{t} h[u]\, du \to \left(\int_0^{t1} h[u]\, du + \int_{t1}^{t} h[u]\, du\right)$$

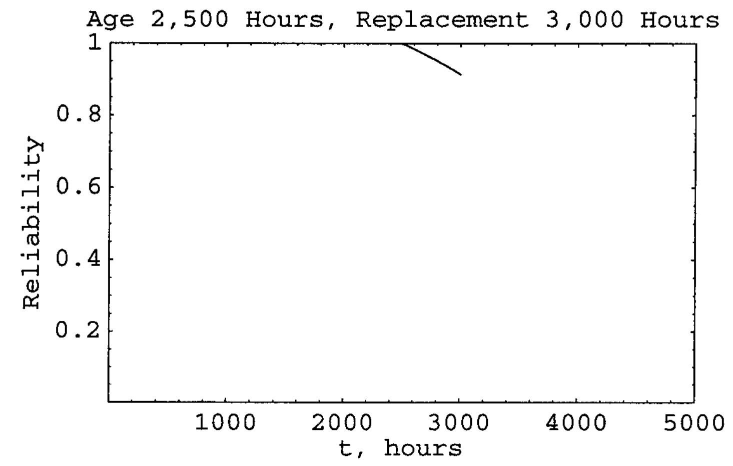
$$e^{-\int_{t1}^{t} h[u]\, du}$$

This is a simplified formula for calculating conditional reliability of a component based on its hazard function. For our hypothetical component, one might define a conditional reliability function thus:

$$\text{condRel}[t1\_,\ t\_] := e^{-\int_{t1}^{t} \text{wblhaz}[u]\, du}$$

The conditional reliability (i.e., probability of not having an accident) is plotted thus:

```
Plot[Evaluate[condRel[2500, t]], {t, 2500, 3000},
  PlotRange → {{0, 5000}, {0, 1}}, Axes → False,
  Frame → True, FrameLabel → {"t, hours", "Reliability",
    "Age 2,500 Hours, Replacement 3,000 Hours", None},
  PlotStyle → RGBColor[0, 1, 0]];
```

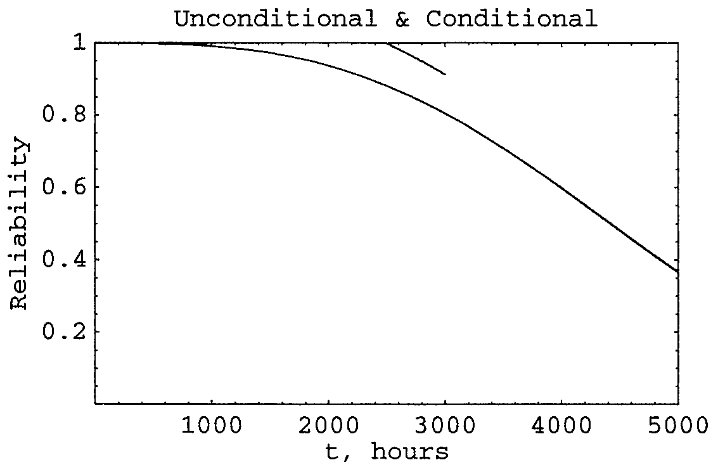


A plot of the unconditional reliability (i.e., the reliability of a new component) is generated but temporarily suppressed thus:

```
Plot[1 - CDF[wbldist, t], {t, 0, 5000},
  PlotRange → {{0, 5000}, {0, 1}}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", "Reliability", "Unconditional", None},
  PlotStyle → RGBColor[0, 0, 1], DisplayFunction → Identity];
```

The conditional and unconditional reliability are now displayed on the same plot:

```
Show[%, %%, FrameLabel →
   {"t, hours", "Reliability", "Unconditional & Conditional", None},
  DisplayFunction → $DisplayFunction];
```



It can be seen from the plot above that using the unconditional reliability function for calculating the probability of a component that is already in-service surviving until replacement can be conservative. The probability of this component surviving until replacement is:

```
condRel[2500, 3000] // N

0.913018
```

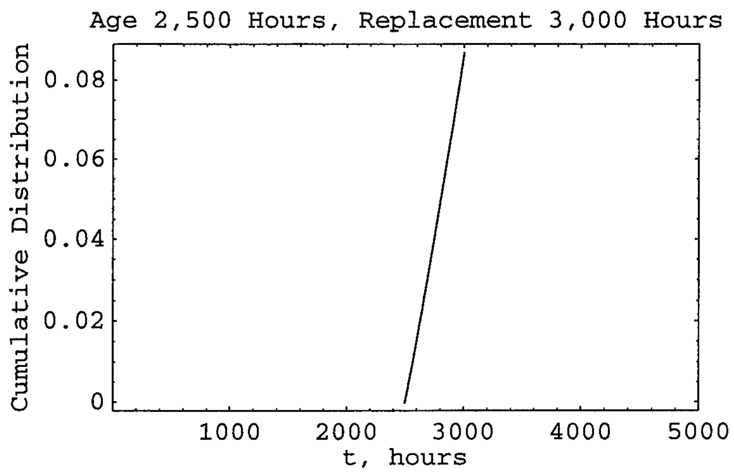
The probability of a new component surviving until replacement is:

```
1 - CDF[wbldist, 3000] // N

0.805735
```

In this case, the conditional and unconditional reliability values are quite different. Not taking credit for components that have survived through an appreciable portion of their life can be rather conservative.

A plot of the conditional cumulative probability of having an accident (i.e., given the current age is 2,500 hours and replacement will occur at 3,000 hours) is:

```
Plot[1 - Evaluate[condRel[2500, t]], {t, 2500, 3000},
  PlotRange → {{0, 5000}, Automatic}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", "Cumulative Distribution",
    "Age 2,500 Hours, Replacement 3,000 Hours", None},
  PlotStyle → RGBColor[0, 1, 0]];
```

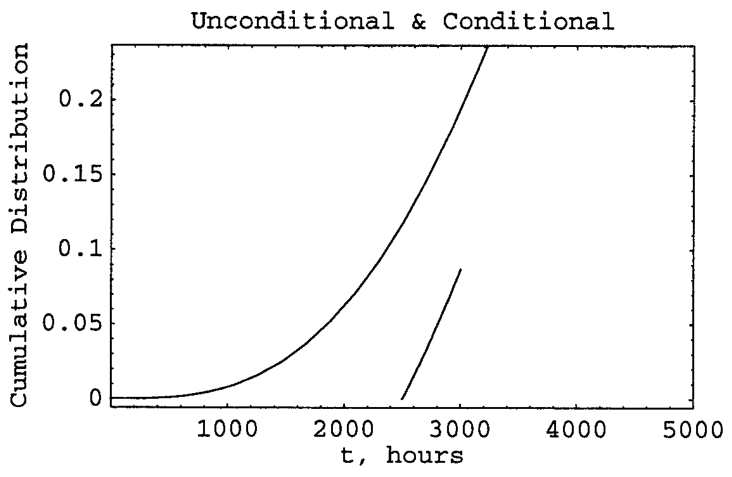


A plot of the unconditional cumulative probability of having an accident is generated but temporarily suppressed:

```
Plot[CDF[wbldist, t], {t, 0, 5000}, PlotRange → {{0, 5000}, Automatic},
  Axes → False, Frame → True, FrameLabel →
   {"t, hours", "Cumulative Distribution", "Unconditional", None},
  PlotStyle → RGBColor[0, 0, 1], DisplayFunction → Identity];
```

The plots of the conditional and unconditional cumulative accident probabilities are now displayed together:

```
Show[%, %%, FrameLabel → {"t, hours",
    "Cumulative Distribution", "Unconditional & Conditional", None},
  DisplayFunction → $DisplayFunction];
```



Once again, we see that calculating an accident probability from the unconditional function for a component already in-service is conservative.

Component-Socket

Let's consider a second hypothetical component installed on the same helicopter as the first. The risk associated with this component has been categorized such that an SSRA has been written for this component, as well as any and all replacements for it. The time-to-accident distribution for this component is modeled with a two-parameter Weibull distribution that has a shape parameter of 2.5 and a scale parameter of 10,000 hours. This distribution is assigned as the value of the symbol *wbldist2* for convenience:

```
wbldist2 = WeibullDistribution[Rationalize[2.5], 10000]
```

$\texttt{WeibullDistribution}\left[\frac{5}{2}, 10000\right]$

The built-in function `Rationalize` was used to convert the value of the shape parameter from an approximate real number to an exact rational number so that *Mathematica* will not use approximations until the analyst insists. The accumulation of numerical errors during a sequence of intermediate steps will be avoided.

The current age of the component is 1,400 hours and when its age reaches 2,200 hours, it will be replaced by a new component with the same time-to-accident distribution. Such replacements will continue as long as the helicopter is in the fleet. This sequence of components installed in/to be installed in a "socket" will be termed a "component-socket".

The hazard function for the current component in this component-socket is:

$$\texttt{wblhaz2[t\_ /; 1400} \le \texttt{t} < \texttt{2200] :=} \frac{\texttt{PDF[wbldist2, t]}}{\texttt{1 - CDF[wbldist2, t]}}$$

The definition above will only apply when $1400 \le t < 2200$. The hazard functions for the next several components that should occupy this socket are:

$$\texttt{wblhaz2[t\_ /; 2200} \le \texttt{t} < \texttt{2 2200] :=} \frac{\texttt{PDF[wbldist2, t - 2200]}}{\texttt{1 - CDF[wbldist2, t - 2200]}}$$

$$\texttt{wblhaz2[t\_ /; 2 2200} \le \texttt{t} < \texttt{3 2200] :=} \frac{\texttt{PDF[wbldist2, t - 2 2200]}}{\texttt{1 - CDF[wbldist2, t - 2 2200]}}$$

$$\texttt{wblhaz2[t\_ /; 3 2200} \le \texttt{t} < \texttt{4 2200] :=} \frac{\texttt{PDF[wbldist2, t - 3 2200]}}{\texttt{1 - CDF[wbldist2, t - 3 2200]}}$$

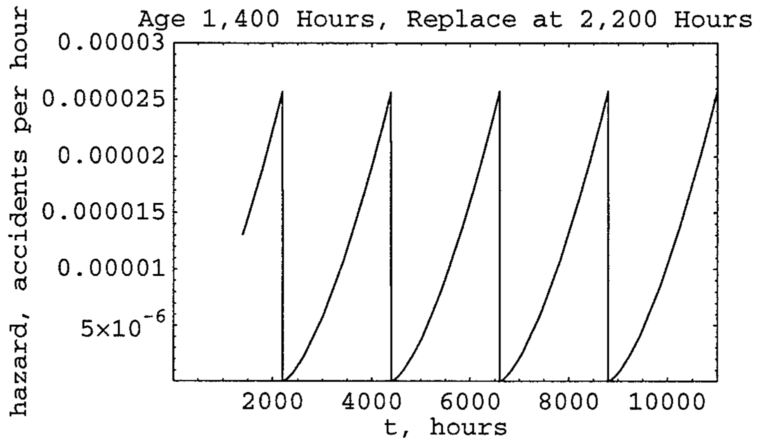
$$\texttt{wblhaz2[t\_ /; 4 2200} \le \texttt{t} < \texttt{5 2200] :=} \frac{\texttt{PDF[wbldist2, t - 4 2200]}}{\texttt{1 - CDF[wbldist2, t - 4 2200]}}$$

It is also convenient to explicitly define the hazard function to be 0 when $0 \le t < 1400$:

```
wblhaz2[t_ /; 0 ≤ t < 1400] := 0
```

A plot of the hazard function for the component-socket is:

```
Plot[wblhaz2[t], {t, 1400, 5 2200},
  PlotRange → {{0, 5 2200}, {0, 0.00003}}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", "hazard, accidents per hour",
    "Age 1,400 Hours, Replace at 2,200 Hours", None},
  PlotStyle → RGBColor[0, 1, 0]];
```



Let's calculate and plot the probability at various points in the future, given the age of the current component is 1,400 hours, that no component in the socket will fail. Calculation of the reliability requires integration of the hazard function above using the formula Exp[-$\int_{t1}^{t} h(u)\,du$] derived earlier. Let's see if *Mathematica* can integrate the component-socket hazard function symbolically:

$$e^{-\int_{1400}^{4000} \text{wblhaz2}[u]\, du}$$

$$e^{-\int_{1400}^{4000} \text{wblhaz2}[u]\, du}$$

Mathematica fails to do so because the hazard was defined as a piecewise-continuous function. The hazard function can be numerically integrated from 1,400 hours until 6,400 hours thus:

```
Exp[-NIntegrate[wblhaz2[u],
   {u, 1400, 2200, 2 2200, 5000 + 1400}, WorkingPrecision → 22]]

0.94557873599601
```

At each of the replacement times, sharp features or singularities are present in the hazard function. Numerical-integration algorithms have difficultly with singularities. For this reason, these replacement times were explicitly provided to `NIntegrate` between the upper and lower integration limits so that so that *Mathematica* will look for the singularities and subdivide the integration region as necessary.

It would be helpful to define a function which will provide an integration-limit list suitable for use with `NIntegrate` with the appropriate replacement times included. First a list of replacement times is needed:

```
replacements = {1400, 2200, 2 2200, 3 2200, 4 2200, 5 2200};
```

A function that will construct an integration-limit list with the appropriate singularities inserted between is:

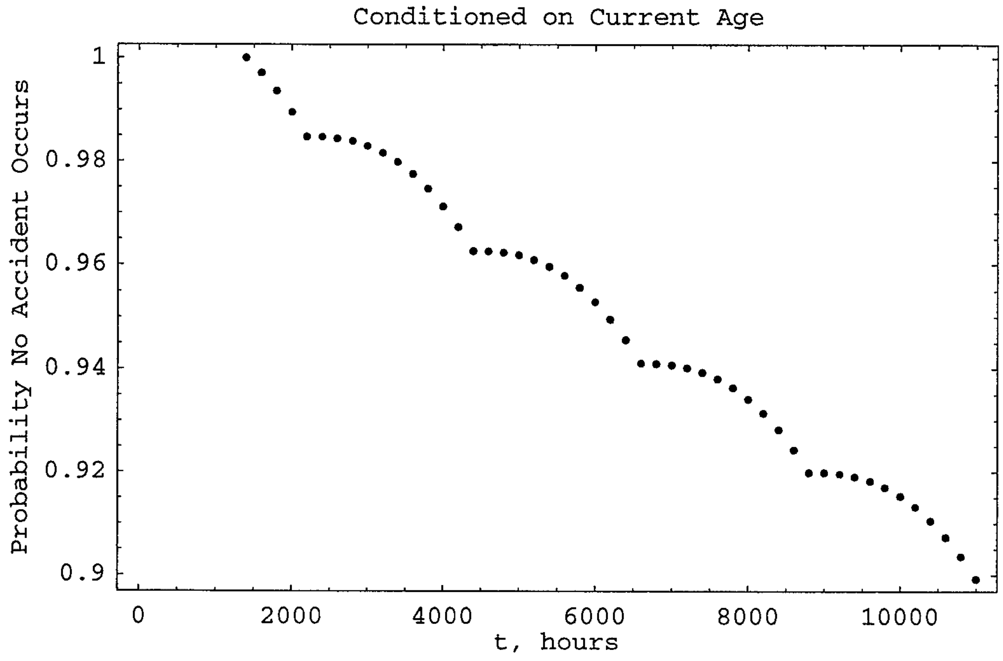
```
intgList[lower_, upper_] :=
 Flatten[{u, lower, Select[replacements, (lower < # < upper) &], upper}]
```

A list of conditional probabilities that an accident will not occur due to the failure of a component in this socket can be generated thus:

```
Short[condRelList = Table[{upperlimit,
    Exp[-NIntegrate[wblhaz2[u], Evaluate[intgList[1400, upperlimit]],
       WorkingPrecision → 22]]}, {upperlimit, 1400, 5 2200, 200}], 10]

{{1400, 1.00000000000000000000},
 {1600, 0.99709786782903257578425 88}, {1800, 0.9936080089001005273334},
 {2000, 0.9895006121025098144}, {2200, 0.984749520753462223},
 {2400, 0.984693816483920}, <<38>>, {10200, 0.91319444339244},
 {10400, 0.91054423241993}, {10600, 0.90735731263780},
 {10800, 0.90360646070543}, {11000, 0.89926779048543}}
```

The list above was assigned as the value of the symbol *condRelList* so that it can be easily referenced. These values can be plotted:

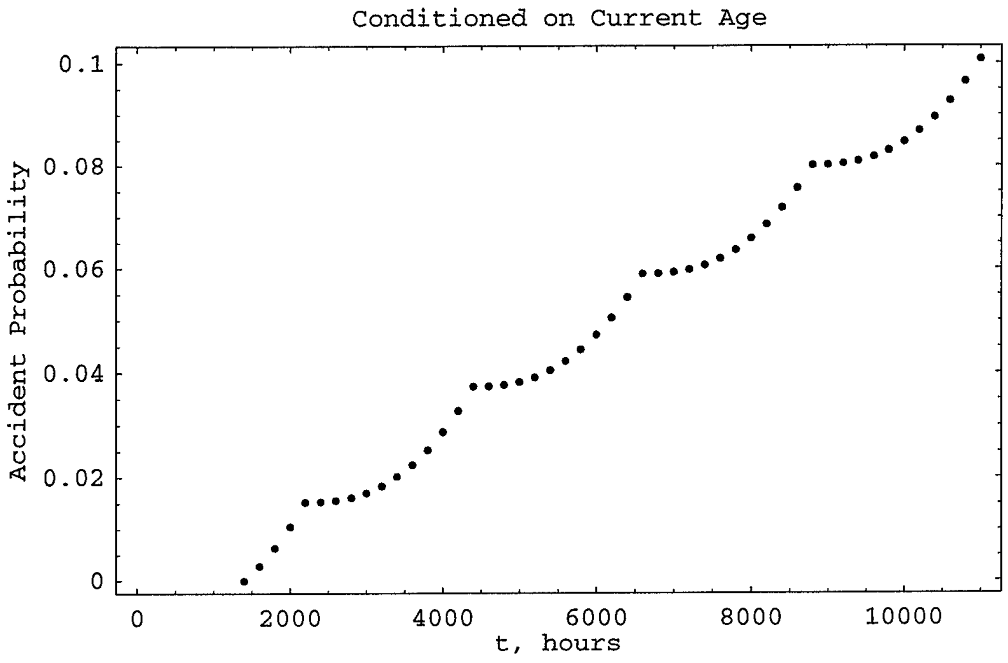
```
ListPlot[condRelList, Frame → True,
  FrameLabel → {"t, hours", "Probability No Accident Occurs",
    "Conditioned on Current Age", None}, Axes → False];
```



A "cusping" effect can be observed in the plot above. A cusp occurs when each component is replaced. When each new component is installed in the socket, the reliability declines more gradually with time than just before the replacement. This is the benefit of replacing aging components before they fail.

A plot of accident probabilities can be obtained by subtracting the probability values in the list *condRel-List* from one and then using `ListPlot` as before:

```
ListPlot[Evaluate[condRelList /. {t_Integer, r_} → {t, 1 - r}],
  Frame → True, FrameLabel → {"t, hours", "Accident Probability",
    "Conditioned on Current Age", None}, Axes → False];
```



Once again the cusping effect is observed.

Aggregate

If we assume that the components are in series (i.e., a severity-level *i* accident will occur if any of the SSRA components fails) and that the failure events are statistically independent, then their hazard functions can be added resulting in an aggregate SSRA hazard function for the helicopter. Theoretical proofs of this are widely available in the literature. A recent reference is Barlow [1998, pp. 138-139]. While considerable redundancy is incorporated in helicopters, SSRAs are typically written for situations where the occurrence of a failure will result in an accident. The series assumption does therefore not appear to be unreasonable for aggregating SSRAs. Meeker and Escobar [1998, pp. 373-374] assert that the independence assumption is conservative for typical physical systems.

We need to put the hazard functions for both the first component and the component-socket on the same time scale. We'll use the age of the helicopter the components are installed on and assume that the current age is 4,200 hours. This corresponds to 2,500 hours for the first component and 1,400 hours for the component-socket. Since the first component is removed at 3,000 hours, we'll define its hazard function to be zero thereafter.

```
wblhaz[t_ /; t > 3000] := 0
```

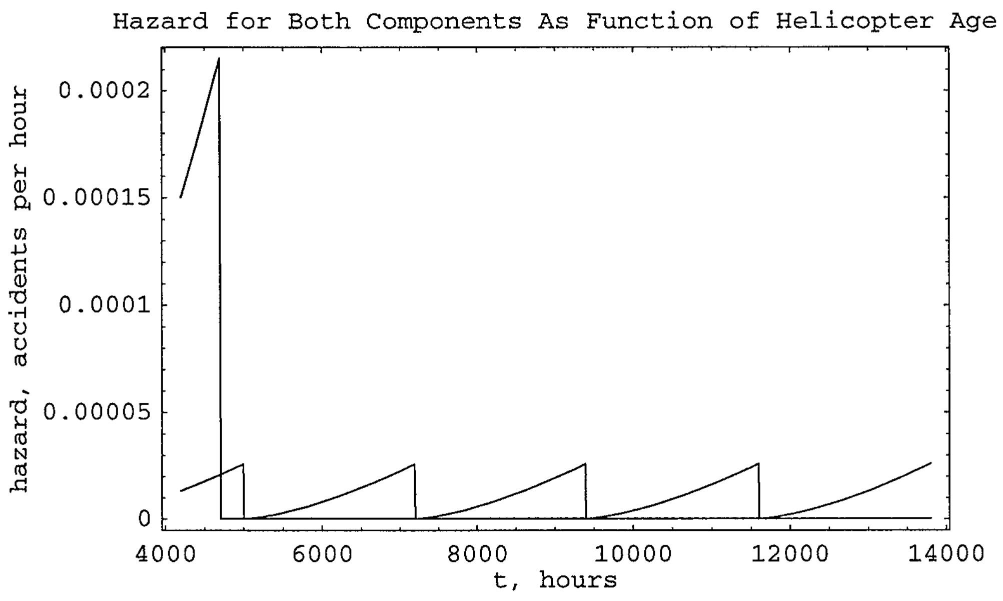
In order to plot the hazard functions on the helicopter time scale, an offset that translates each of the hazard function time scales to the helicopter time scale is needed. The offsets for the first component and the component-socket hazard functions, respectively, are:

```
offset1 = 4200 - 2500;
```

```
offset2 = 4200 - 1400;
```

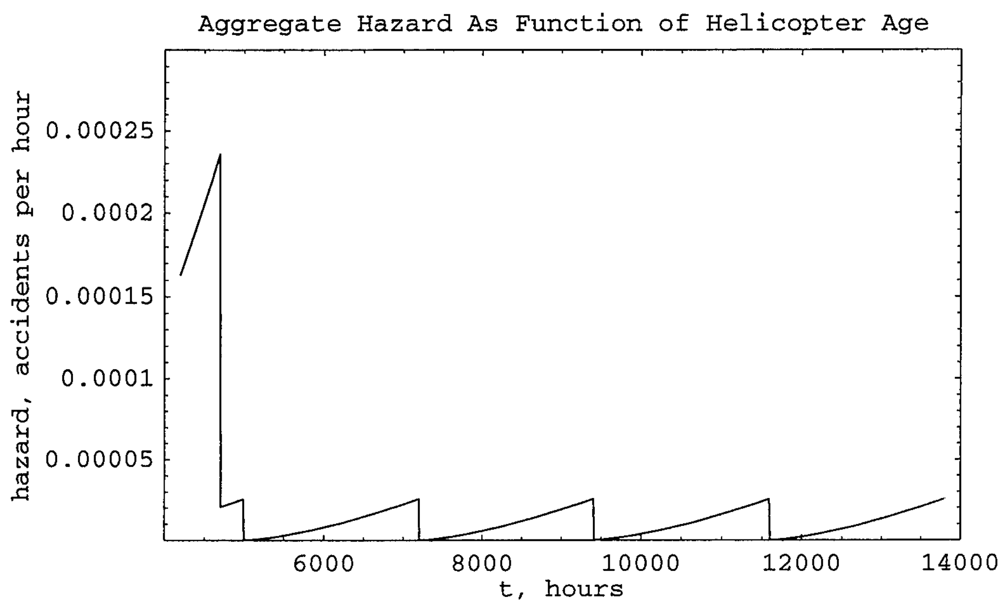
The hazard functions for the components are plotted together as a function of helicopter age:

```
Plot[{wblhaz[t - offset1], wblhaz2[t - offset2]},
  {t, 4200, 4200 + (5 2200 - 1400)}, PlotRange → All, Axes → False,
  Frame → True, FrameLabel → {"t, hours", "hazard, accidents per hour",
    "Hazard for Both Components As Function of Helicopter Age", None},
  PlotStyle → {RGBColor[0, 0, 1], RGBColor[0, 1, 0]}];
```



The hazard function of the first component is larger in magnitude but shorter in duration than that of the component-socket. The sum of the component hazard functions is plotted together as a function of helicopter age:

```
Plot[wblhaz[t - offset1] + wblhaz2[t - offset2],
  {t, 4200, 4200 + (5 2200 - 1400)},
  PlotRange → {{4000, 14000}, {0, .0003}}, Axes → False, Frame → True,
  FrameLabel → {"t, hours", "hazard, accidents per hour",
    "Aggregate Hazard As Function of Helicopter Age", None},
  PlotStyle → {RGBColor[0, 0, 1], RGBColor[0, 1, 0]}];
```



In order to calculate and plot the probability that an accident will not occur, the formula Exp[-$\int_{t1}^{t} h(u)\, du$] will be used again. As before, the sharp features in the hazard functions will need to be explicitly provided to `NIntegrate`. The replacement times for the component-socket need to be translated to helicopter hours by adding the appropriate offset to each replacement time thus:

```
replacements = replacements + offset2

{4200, 5000, 7200, 9400, 11600, 13800}
```

The removal time for the first component, translated with the appropriate offset, is included as well:

```
replacements = Sort[Prepend[replacements, 3000 + offset1]]

{4200, 4700, 5000, 7200, 9400, 11600, 13800}
```

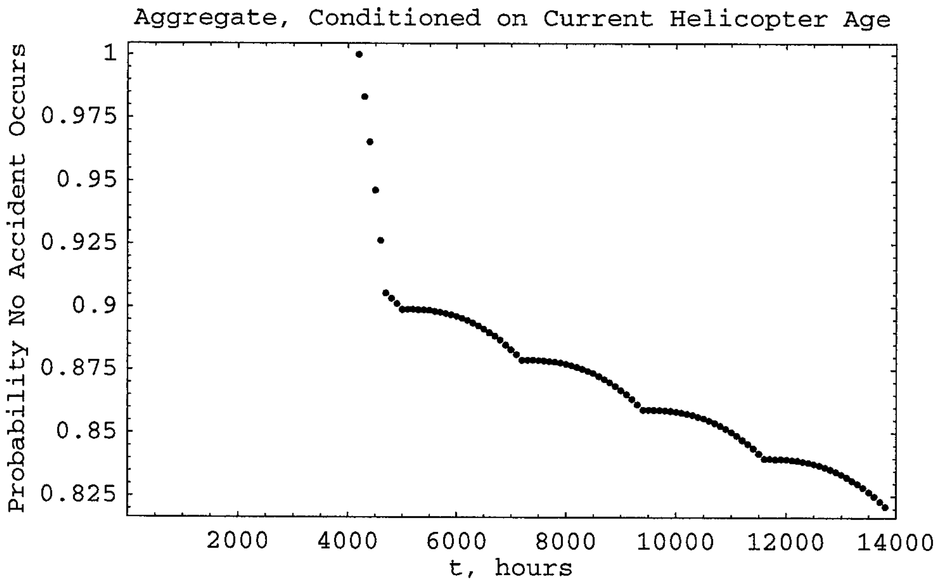
A list of conditional probabilities that an accident will not occur on the helicopter due to either the first component or the component-socket.

```
Short[condRelHelo = Table[
   {upperlimit, Exp[-NIntegrate[wblhaz[u - offset1] + wblhaz2[u - offset2],
       Evaluate[intgList[4200, upperlimit]], WorkingPrecision → 22]]},
   {upperlimit, 4200, 4200 + (5 2200 - 1400), 100}], 10]
```

```
{{4200, 1.000000000000000000000},
 {4300, 0.9831549278803436394419836}, {4400, 0.9652478690352440697078427},
 {4500, 0.9462976424743834312074 6}, {4600, 0.926330212113591412776 3},
 {4700, 0.905378693718924717 19}, <<86>>, {13400, 0.82843329655228},
 {13500, 0.82678679227003}, {13600, 0.82500870230706},
 {13700, 0.82309639446500}, {13800, 0.82104741955445}}
```

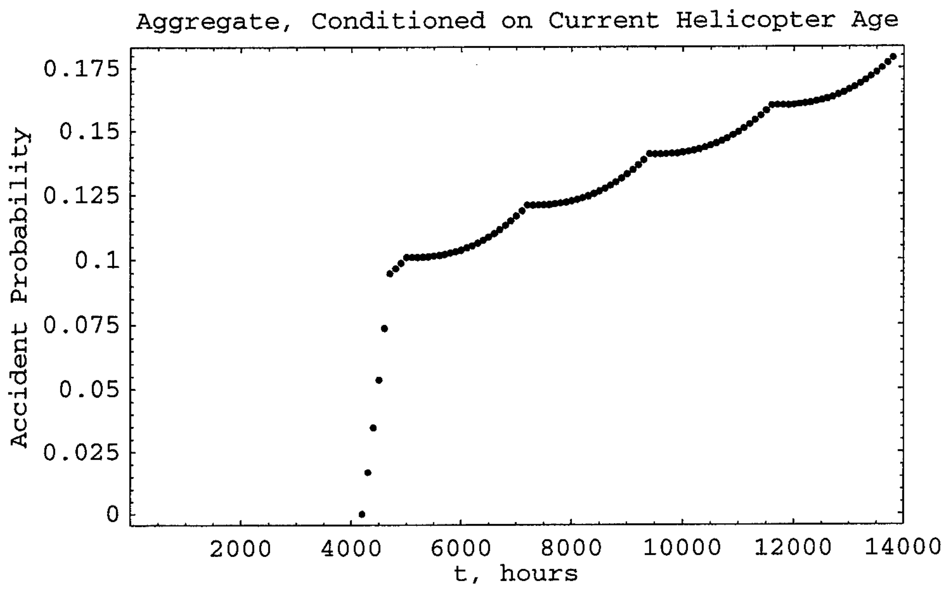
The reliability data are now plotted:

```
ListPlot[condRelHelo, Frame → True,
  FrameLabel → {"t, hours", "Probability No Accident Occurs",
    "Aggregate, Conditioned on Current Helicopter Age", None},
  Axes → False, PlotRange → {{0, 14000}, Automatic}];
```



A plot of accident probabilities can be obtained by subtracting the probability values in the list *condRel-Helo* from one and then using `ListPlot`:

```
ListPlot[Evaluate[condRelHelo /. {t_Integer, r_} → {t, 1 - r}],
  Frame → True, FrameLabel → {"t, hours", "Accident Probability",
    "Aggregate, Conditioned on Current Helicopter Age", None},
  Axes → False, PlotRange → {{0, 14000}, Automatic}];
```



It may also be of interest to calculate the number of accidents expected during a specified time period (e.g., in the next 2,000 flight hours). During any time period of interest, a helicopter may or may not have an accident. This may be viewed as a Bernoulli trial. The probability that a helicopter will have an accident can be calculated as above. This probability is also the mean or expected number of accidents for the Bernoulli distribution. In the next 2,000 hours, for example, the accident probability and the expected number of accidents for our hypothetical helicopter is:

```
1 - Exp[-NIntegrate[wblhaz[u - offset1] + wblhaz2[u - offset2],
    Evaluate[intgList[4200, 4200 + 2000]], WorkingPrecision → 22]]

0.1053800342213
```

The expected number of accidents for each helicopter in a fleet could be calculated in a similar fashion and then summed in order to obtain the number of accidents expected in the fleet during the time period of interest.

Expected-Accident Aggregation (EAA) Model

The EAA model aggregates the expected number of yearly accidents for each SSRA over a multi-year time horizon. This model addresses only fleet-level risk, not the individual risk for each helicopter. The data requirements for this model are, however, within reach of the current state of data collection. AMSAA has developed two software implementations of this model, one in *Mathematica* and the other in *Excel.* The software generates tables and graphs of expected accidents as a function of calendar time and the next *n* flight hours. The software also calculates and plots accident probabilities.

■ Theoretical Description

The definition of the homogeneous Poisson process (HPP) can be phrased, with respect to an SSRA, thus: If accidents due to SSRA *i* occur successively in time with intervals between accidents independent and identically distributed according to the exponential distribution, then the number of accidents in an interval is a Poisson process with mean equal to the occurrence rate times the interval length. (This definition is a re-phrasing of one found in Barlow and Proschan [1981, p. 63].) The assumption being made is, in effect, that an accident is as likely to occur during one interval of time as any other interval of equal size. One objection to this assumption is that the intervals between accidents may not exactly be identically distributed. One reason is that accidents that result in the loss of a helicopter change the fleet size. Fleet size and flight-hours per helicopter also change over time for other reasons (e.g., delivery of additional helicopters, changes in operational tempo). In order to make this assumption reasonable, it is recommended that the mean for an SSRA HPP be defined for an appropriate calendar-time interval. For the Apache fleet, HPP parameters were defined for periods of one calendar year [Cushing 2001]. With a constraint such as this imposed, the assumption of independent intervals appears to be reasonable. This is the first assumption made in the EAA model.

For an HPP, the number of accidents between time points *r* and *s*, where $r < s$, follows the Poisson distribution. Functions for the Poisson distribution are contained in the standard add-on package *Statistics`DiscreteDistributions`*, which is loaded thus:

```
Needs["Statistics`DiscreteDistributions`"]
```

The usage message for the Poisson distribution is:

```
?PoissonDistribution

PoissonDistribution[mu]
  represents the Poisson distribution with mean mu.
```

Since the mean is the accident occurrence rate times the interval length, the probability of x accidents is:

```
PDF[PoissonDistribution[OccurrenceRate (r - s)], x]
```

$$\frac{e^{-\text{OccurrenceRate}\,(r-s)}\,(\text{OccurrenceRate}\,(r-s))^{x}}{x!}$$

If each SSRA is modeled with a HPP, then they can be aggregated by superposition. The superposition of independent HPPs with expected number of accidents (means) $\mu_1, \mu_2, \ldots \mu_n$, is also an HPP with aggregate expected number of accidents $\mu = \mu_1 + \mu_2 + \ldots + \mu_n$. (One may refer to Barlow and Proschan [1981, p. 68] for a proof of this theorem.) Independent SSRAs is a second assumption of the EAA model. In essence, it is assumed that the number of accidents in a time interval due to one SSRA is independent of the number of accidents due to another SSRA. This is not the same as, and should not be confused with, the assumption that a system contains independent components. No significant issues with the assumption of independent SSRAs have been identified thus far.

The theory behind the EAA model is graphically depicted in the figure below.

Expected Accident Aggregation (EAA) Model

HPP = Homogeneous Poisson Process

parameter μ_{ij} = expected yearly accidents in fleet for *j*th year of *i*th SSRA (includes effects of SSRA attrition, changes in fleet size, flying-hour profiles)

Assumptions:

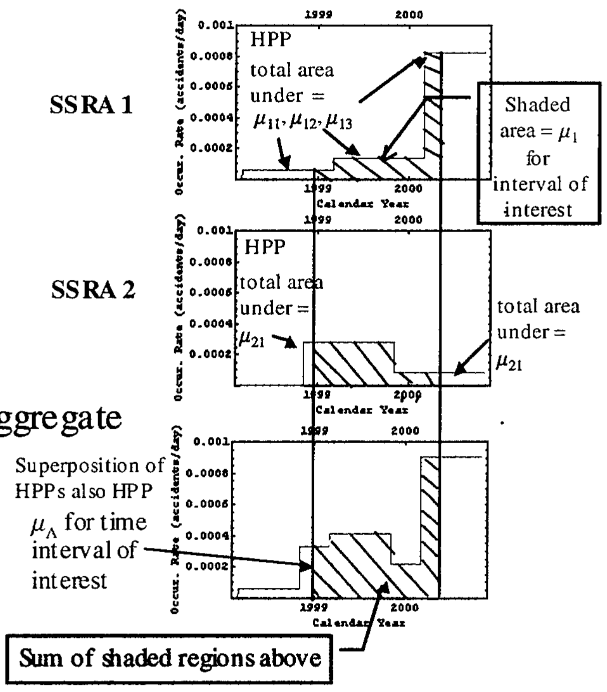
1. Accident from SSRA *i* as likely to occur in one portion of year as any other equally-sized portion of same year (calculate constant occurrence rate for each μ_{ij})
2. SSRA HPPs are independent

SSRA Aggregate

EAA model software generates tables & graphs of expected accident as function of calendar time and next *n* flight hours

Poisson distribution to used to calculate accident probabilities

$$\text{Prob of x accidents} = \frac{e^{-\mu}\,\mu^{x}}{x!}$$



- **Calendar Time Versus Flight Hours**

The primary input that the SSRA subject matter expert provides to the EAA model is in terms of the expected number of accidents in a calendar-time period. Calendar time was chosen for several reasons:

Reason #1: Some factors relevant to SSRA risk predictions are driven by the passage of calendar time, not the accumulation of flight hours. One example is that the procurement and manufacture of new components intended to replace weak, currently-installed components requires a certain amount of calendar time. The number hours the fleet is flying is irrelevant. It is certainly the case that SSRA risk predictions are also driven by the quantity of helicopters in the fleet and the number of flight hours they accumulate. Since the EAA model requires risk-predictions based on calendar time, the responsibility for considering the impact of both calendar time and flight hours on an SSRA rests properly with the SSRA subject matter expert, not with the analyst aggregating the SSRAs.

Reason #2: It is easy to understand what the EAA model risk prediction input is -- the number of accidents expected in a defined calendar-time period. Accident rates expressed, for example, on a per 100,000 flight hours basis have multiple interpretations (e.g., a hazard function, a ratio of a quantity of accidents to a quantity of flight hours).

Reason #3: The assumptions made by the aggregation model are few and weak. Additional assumptions may be required if risk predictions were expressed in terms of flight hours. To avoid additional assumptions, risk predictions based on flight hours could be provided for specific calendar time periods since calendar time does have an impact. This then is no easier for the SSRA experts than expected accidents in a calendar time period.

Reason #4: Complete latitude is given to the SSRA engineers to improve their risk predictions. No model or modeling assumptions flow down to them.

There is interest in both accidents as a function of calendar time and as a function of flight hours. The former is good for considering the accident quantities and probabilities in calendar-time intervals (e.g., the number of severity-level I accidents expected to occur in a specific helicopter fleet in calendar year 2002). The latter is preferred when comparing large and small fleets. The EAA model primarily provides results as a function of calendar time but also provides key results as a function of the next *x* flight hours.

- **EAA Model Functions**

For the *Mathematica* implementation of the EAA model, 21 new functions were developed. These functions were collected together in a *Mathematica* package `AviationSafety`ExpectedAccidentAggregationModel`` which is loaded thus:

```
Needs["AviationSafety`ExpectedAccidentAggregationModel`"]
```

The current version of the software is:

```
?ExpectedAccidentAggregationModel

ExpectedAccidentAggregationModel.m (version 1.0.1) is a package that
  contains functions for the Expected Accident Aggregation model.
```

The package may be found at Appendix A, which includes the *Mathematica* code for each of the new functions as well as a discussion of the algorithms used. Appendix B contains a listing of the usage messages for each of the new functions, a handy reference for the user. The use of these new functions, in combination with the built-in *Mathematica* functions, will be demonstrated on notional data in Chapters 3 and 4.

A short description of the new functions may be helpful. Five of the new functions address data input and manipulation:

- `FirstSSRA` searches a list of SSRAs and returns the one that began first.
- `LastSSRA` searches a list of SSRAs and returns the one that began last.
- `SSRASelect` searches a list of SSRAs and returns a list of those with a specified severity level.
- `SSRATest` tests a list of SSRA data to ensure it's structured correctly and doesn't contain obvious errors such as dates not in chronological order.
- `FlightHoursTest` tests a list of flight-hours data to ensure it's structured correctly and doesn't contain obvious errors such as dates not in chronological order.

Five of the new functions address accident occurrence rates:

- `OccurrenceRate` calculates the occurrence rate, in accidents per day, on a specified date for a specific SSRA.
- `OccurrenceRatePie` generates a pie chart of occurrence rates, in accidents per day, on a specified date for all the SSRAs.
- `OccurrenceRatePlot` plots, from one date to another, the occurrence rates (accidents per day) for all the SSRAs, or just a subset of them.
- `OccurrenceRateTable` generates a table of occurrence rates, in accidents per day, on a specified date for all the SSRAs, or just a subset of them.
- `AggregateOccurrenceRatePlot` plots, from one date to another, the aggregate occurrence rate (accidents per day) for all the SSRAs, or just a subset of them.

Seven of the new functions address the expected number of accidents as a function of calendar time:

- `ExpectedAccidents` calculates the expected number of accidents from one date to another for a specific SSRA.

- `ExpectedAccidentsPie` generates a pie chart of the expected number of accidents from one date to another for all the SSRAs.
- `ExpectedAccidentsPlot` plots, as a function of calendar time, the expected number of accidents from a starting date forward for all the SSRAs, or just a subset of them.
- `ExpectedAccidentsTable` generates a table of *n*-year expected accidents from a starting date forward for all the SSRAs, or just a subset of them.
- `AggregateExpectedAccidents` calculates the aggregate expected number of accidents from one date to another for all the SSRAs, or just a subset of them.
- `AggregateExpectedAccidentsPlot` plots, as a function of calendar time, the aggregate expected number of accidents from a starting date forward for all the SSRAs, or just a subset of them.
- `ExpectedAccidentsTrendPlot` plots, as a function of calendar time, the *n*-year aggregate expected number of accidents for all the SSRAs.

Four of the new functions address the expected number of accidents as a function of the next x flight hours:

- `FlightHours` calculates the number of flight hours predicted to be accrued between one date and another.
- `FlightHoursDate` calculates, starting from an initial date, the date at which a specified number of flight hours will be accrued.
- `ExpectedAccidentsNext` calculates, starting from an initial date, the aggregate number of accidents expected to occur in the next n flight hours for all the SSRAs.
- `ExpectedAccidentsNextPlot` plots, as a function of calendar time, the aggregate number of accidents expected to occur in the next n flight hours for all the SSRAs.

The functions listed above may be used to analyze SSRAs from various severity levels both separately and in combination (but with equal weighting). Many graphs and tables may be generated for each of the severity levels and combinations thereof. Tables and pie charts may be generated for the accident occurrence rates at present. Individual and aggregate SSRA occurrence rates can be plotted to show their incremental risk over time. Tables and pie charts can be generated for the expected number of accidents at particular points in time. The expected number of accidents may be aggregated over various time periods and presented in a variety of ways. Plots of the next year and next 100,000 flight hours expected-accidents trends may also generated and compared. The impacts of the most recent SSRA and fixing the worst SSRAs may be considered as well. Accident probabilities can be calculated and plotted using a mix of built-in and add-on functions already available in *Mathematica*. The various graphics generated with the EAA model help one put an SSRA into context, not just now, but over time. How the "riskiness" of an SSRA changes with time will be illustrated with the notional example in Chapters 3 and 4. It's important to see an SSRA within the context of all the SSRAs for a given helicopter fleet over time. An SSRA that may appear to be a top priority at present may be eclipsed when one looks a couple of years into the future. This will be illustrated in Chapters 3 and 4 with a notional example.

■ EAA Model Process

The use of these new functions, in combination with the built-in *Mathematica* functions, will be illustrated on notional data in Chapters 3 and 4. Chapter 3 will address data requirements and input for the EAA model, whereas Chapter 4 will address the analysis.

■ Model Benefits

Application of the EAA model requires that a few beneficial changes be made to the SSRA process:

- All SSRAs must be quantitative. Previously, some SSRAs did not produce quantitative accident predictions. Instead, an SSRA was placed in a qualitative category such as "improbable" or "remote".
- All SSRAs must provide a standardized quantitative prediction in terms of expected accident quantities for specified calendar-time intervals (e.g., 0.7 severity-level I accidents expected from helicopter fleet XYZ between 5 Jan 2002 and 5 Jan 2003).
- SSRA predictions should change over time as appropriate. Previously, SSRA accident predictions did not appear to be time-dependent.

The necessary changes were made to the SSRAs for the Apache fleet when the EAA model was applied to it. This application is documented in AMSAA Technical Report 695 [Cushing 2001]. An additional enhancement implied by the bullets above is that the category an accident prediction is placed in should change as warranted by changes in short- or mid-term risks. For example, an SSRA prediction may initially be place in "Hazard Probability (Frequency)" category C, but when sufficient risk mitigation occurs, it is downgraded to category D.

Key benefits of the EAA model are:

- it depicts risks through time,
- it aggregates risks from individual SSRAs in a variety of ways,
- its graphics show progress in terms of resolving SSRAs and can be used as an effective management tool.

Summary

To support the Aviation Safety IPT, the AMRDEC asked AMSAA to develop a model that aggregates the risks due to the collection of open SSRAs applicable to a helicopter fleet at a point in time. AMSAA developed two such models, the Component-Socket Hazard model and the EAA model. These models can be used to calculate accident probabilities due to SSRA causes as SSRAs are identified and resolved through time. This will allow decision makers to consider the cumulative risk to the fleet due to all open SSRAs when deciding which risk-mitigation option to select for a new SSRA. The two models were presented in this chapter.

The Component-Socket Hazard model, aggregates component hazard functions and can assess risk at the fleet level and also provide the risk associated with each helicopter tail number. This model can address helicopter-to-helicopter variability and will calculate the risks at any point in the life-cycle. However, the model requires component aging models and component-age data. Component-aging models and component-age data are not currently available for all SSRAs. A rudimentary implementation of this model in *Mathematica* may be found in this chapter. It may be possible to apply this model in the future when component tracking data becomes available.

The EAA model aggregates the expected number of yearly accidents for each SSRA over a multi-year time horizon. This model addresses only fleet level risk, and not the individual risk for each helicopter. However, the data requirements for this model match the current state of data collection. AMSAA has developed two implementations of this model, one in *Mathematica* and the other in *Excel.* The *Mathematica* functions for the EAA model may be found in Appendix A. The use of these new functions, in combination with the built-in *Mathematica* functions, will be demonstrated on notional data in Chapters 3 and 4. With the EAA model, the expected number of accidents may be aggregated over various time periods and presented in a variety of ways. Plots of the next year and next 100,000 flight hours expected-accidents trends may also generated and compared. The impacts of the most recent SSRA and fixing the worst SSRAs may be considered as well. Accident probabilities can be calculated and plotted using a mix of built-in and add-on functions already available in *Mathematica*. The various graphics generated with the EAA model help one put an SSRA into context, not just now, but over time. How the "riskiness" of an SSRA changes with time will be illustrated with the notional example in Chapters 3 and 4. It's important to see an SSRA within the context of all the SSRAs for a given helicopter fleet over time. An SSRA that may appear to be a top priority at present may be eclipsed when one looks a couple of years into the future.

THIS PAGE INTENTIONALLY LEFT BLANK.

Chapter 3

Input of System Safety Risk Assessment (SSRA) Data

In this chapter, SSRA data for a notional helicopter fleet are structured so that the Expected Accident Aggregation (EAA) model can be used to aggregate and analyze the data in the next chapter. Data structures for the flight-hours data are also developed in this chapter. Flight-hours data will be used to relate the expected numbers of accidents to the flight-hour profile (e.g., calculating the expected number of accidents in the next 100,000 hours). This permits one to adjust and compare the expected numbers of accidents for helicopter fleets of various sizes and usage levels. It is thought that this provides a fair manner in which to compare the risks of differing fleet sizes.

The required inputs for the EAA model are saved to a file for aggregation and analysis in Chapter 4.

Description of Required Data

- **SSRA Data**

The input data for each SSRA consists of the following information:

- SSRA name
- helicopter-fleet applicability
- severity level (I = catastrophic, II = critical, etc.)
- calendar start date for the first expected-accident prediction
- expected fleet accidents predicted between above date and next date
- date expected-accident prediction above ends and next prediction, if there is one, begins

The last two elements above are repeated as necessary. If the SSRA applies to more than one model, a separate estimate as described above should be provided for each. The SSRA data must be structured in a specific manner in order for the new add-on functions that implement the EAA model to work. The data for each SSRA, appropriately structured, is presented in this chapter.

▪ **Data on Flight Hours**

The flight-hours data for each helicopter fleet consists of the following information:

- helicopter-fleet applicability
- calendar start date for the first flight-hours prediction
- prediction of the number of fleet flight hours between above date and next date
- date prediction above ends and next prediction, if there is one, begins

The last two elements above are repeated as necessary. The flight-hours data must be structured in a specific manner in order for the new add-on functions that implement the EAA model to work. The structuring of the flight-hours data is presented after the SSRA data.

Preliminaries

The add-on functions for the EAA model are contained within the new *Mathematica* package found in Appendix A. The package is loaded with the built-in function `Needs` as follows:

```
Needs["AviationSafety`ExpectedAccidentAggregationModel`"]
```

The usage message for this package is:

```
?ExpectedAccidentAggregationModel

ExpectedAccidentAggregationModel.m (version 1.0.1) is a package that
  contains functions for the Expected Accident Aggregation model.
```

The SSRA data will be structured as a list of lists. The symbol *HelicopterSSRAs* will be used for the SSRA data. We start by assigning an empty list as the value of this symbol:

```
HelicopterSSRAs = {};
```

It should be noted that all executable cells in this chapter, except for those that generate graphics or manipulate files, have been designated as initialization cells. As a result, the data for all the SSRAs can be generated automatically. This is very helpful when preparing a large number of SSRAs.

Preparation and Input of SSRA Data

In the interests of brevity, ten notional SSRAs are created here and each SSRA will be assigned to severity-levels I or II.

▪ Clutch

The data for this SSRA are:

- SSRA name: "clutch"
- helicopter-fleet applicability: "Notional Helicopter Fleet"
- severity level: "I"
- expected-accident prediction calendar start date: {1998, 1, 1}
- number of accidents expected between above date and next date: $2\ 10^{-2}$
- date expected-accident prediction above ends: {1998, 7, 1}
- number of accidents expected between above date and next date: $5\ 10^{-2}$
- date expected-accident prediction above ends: {1999, 1, 1}
- number of accidents expected between above date and next date: $3\ 10^{-1}$
- date expected-accident prediction above ends: {1999, 7, 1}
- number of accidents expected between above date and next date: $4\ 10^{-1}$
- date expected-accident prediction above ends: {2000, 1, 1}
- number of accidents expected between above date and next date: $5\ 10^{-1}$
- date expected-accident prediction above ends: {2000, 7, 1}
- number of accidents expected between above date and next date: $6\ 10^{-1}$
- date expected-accident prediction above ends: {2001, 1, 1}
- number of accidents expected between above date and next date: $1\ 10^{-1}$
- date expected-accident prediction above ends: {2001, 7, 1}
- number of accidents expected between above date and next date: $1\ 10^{-2}$
- date expected-accident prediction above ends: {2002, 1, 1}
- number of accidents expected between above date and next date: $1\ 10^{-3}$
- date expected-accident prediction above ends: {2002, 7, 1}
- number of accidents expected between above date and next date: 0
- date expected-accident prediction above ends: {2003, 1, 1}

It is expected that this SSRA will be entirely fixed by the final date above. The data above is structured as follows:

```
clutch = {"clutch", "Notional Helicopter Fleet", "I", {1998, 1, 1},
  2 10^-2, {1998, 7, 1}, 5 10^-2, {1999, 1, 1}, 3 10^-1, {1999, 7, 1}, 4 10^-1,
  {2000, 1, 1}, 5 10^-1, {2000, 7, 1}, 6 10^-1, {2001, 1, 1}, 1 10^-1,
  {2001, 7, 1}, 1 10^-2, {2002, 1, 1}, 1 10^-3, {2002, 7, 1}, 0, {2003, 1, 1}}
```

$\{$clutch, Notional Helicopter Fleet, I, {1998, 1, 1}, $\frac{1}{50}$, {1998, 7, 1}, $\frac{1}{20}$, {1999, 1, 1}, $\frac{3}{10}$, {1999, 7, 1}, $\frac{2}{5}$, {2000, 1, 1}, $\frac{1}{2}$, {2000, 7, 1}, $\frac{3}{5}$, {2001, 1, 1}, $\frac{1}{10}$, {2001, 7, 1}, $\frac{1}{100}$, {2002, 1, 1}, $\frac{1}{1000}$, {2002, 7, 1}, 0, {2003, 1, 1}$\}$

It can be readily seen from the example above that the SSRA data structure is a list consisting of:

- SSRA name (string)
- helicopter-fleet applicability (string)
- severity level I, II, III or IV (string)
- expected-accident prediction calendar start date (a list of integers formatted {yyyy, m, d})
- number of fleet accidents expected between above date and next date (Real, Rational or Integer)
- date expected-accident prediction above ends and next prediction, if there is one, begins (formatted as date above)

Additional expected-accident predictions were appended by repeating the last two elements above.

It should also be noted that the calendar dates for the expected-accident predictions above were always six months apart. Other intervals, even non-uniform intervals, could have been used. For simplicity, all calendar-time intervals for SSRA data in this chapter will be six months.

It would be prudent to test this SSRA to ensure it's structured correctly and doesn't contain obvious errors (e.g., dates not in chronological order) before including it in the list of SSRAs. The new add-on function `SSRATest` was developed for this purpose. The usage message for `SSRATest` is:

```
? SSRATest

SSRATest[ssra] tests an SSRA to ensure it's structured correctly.
```

The data structure is tested thus:

```
SSRATest[clutch]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, clutch];
```

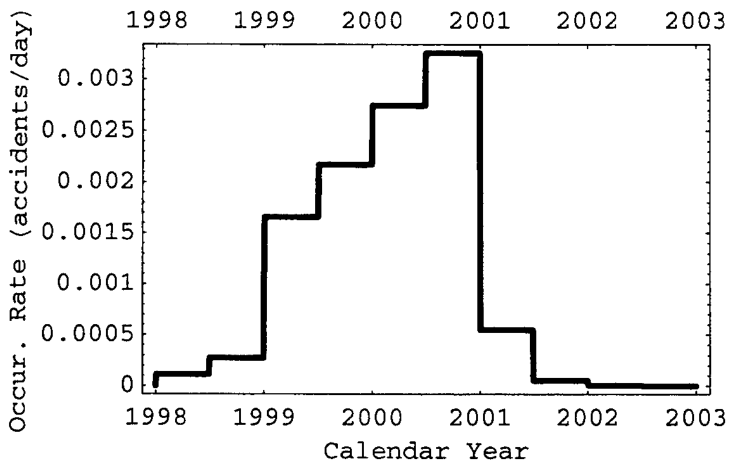
It can be helpful to plot and examine the occurrence rate for an SSRA. The new add-on function `OccurrenceRatePlot` was developed for this purpose. The usage message for this function is:

```
?OccurrenceRatePlot
```

```
OccurrenceRatePlot[SSRAlist, yearInc, opts] plots the occurrence
  rates (accidents per day) of all SSRAs in SSRAlist from a
  day before the earliest begins to a day after the last ends.
  yearInc specifies the calendar-year increments to be used on
  the plot. OccurrenceRatePlot[SSRAlist, {startDate, endDate},
  yearInc, opts] plots the occurrence rates of all SSRAs from
  startDate to endDate. OccurrenceRatePlot[SSRAlist, name, yearInc,
  opts] plots the occurrence rate of the named SSRA from a day
  before it begins to a day after it ends. OccurrenceRatePlot[
  SSRAlist, name, {startDate, endDate}, yearInc, opts] plots the
  occurrence rate of the named SSRA from startDate to endDate.
  OccurrenceRatePlot[SSRAlist, nameList, yearInc, opts] plots the
  occurrence rates of only SSRAs in nameList from a day before the
  earliest begins to a day after the last ends. OccurrenceRatePlot[
  SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots
  the occurrence rates of only SSRAs in nameList from startDate to
  endDate. opts is an optional argument for specifying Plot options.
```

The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{clutch}, 1];
```



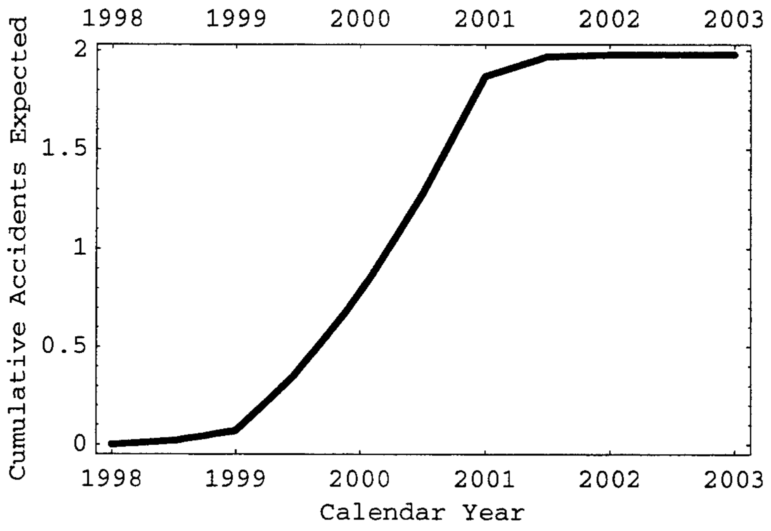
The occurrence rate initially increases over time, due possibly to aging and/or increasing fleet size, and then decreases due to improved risk mitigation, replacement of weak components and/or decreasing fleet size. The occurrence rate is zero at the start of calendar year 2003.

It can also be helpful to plot and examine the cumulative number of accidents expected to occur from the beginning of the SSRA forward. The new add-on function `ExpectedAccidentsPlot` will generate a variety of expected-accident plots. The usage message is:

```
?ExpectedAccidentsPlot

ExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots the expected
  number of accidents for all SSRAs in SSRAlist from a day before
  the earliest begins to a day after the last ends. yearInc
  specifies the calendar-year increments to be used on the plot.
  ExpectedAccidentsPlot[SSRAlist, {startDate, endDate}, yearInc,
  opts] plots the expected number of accidents for all SSRAs from
  startDate to endDate. ExpectedAccidentsPlot[SSRAlist, name,
  yearInc, opts] plots the expected number of accidents for the
  named SSRA from a day before it begins to a day after it ends.
  ExpectedAccidentsPlot[SSRAlist, name, {startDate, endDate},
  yearInc, opts] plots the expected number of accidents for the named
  SSRA from startDate to endDate. ExpectedAccidentsPlot[SSRAlist,
  nameList, yearInc, opts] plots the expected number of accidents
  for only SSRAs in nameList from a day before the earliest begins
  to a day after the last ends. ExpectedAccidentsPlot[SSRAlist,
  nameList, {startDate, endDate}, yearInc, opts] plots the expected
  number of accidents for only SSRAs in nameList from startDate to
  endDate. opts is an optional argument for specifying Plot options.
```

A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{clutch}, 1];
```



This plot initially curves upward since the occurrence rate increases with time then downward when the occurrence rate decreases with time. The plot levels off when the occurrence rate goes to zero. It is readily apparent that the expected-accident curve is linear but the slope changes when the level of the occurrence-rate changes.

■ Solenoid Valve

The data for this SSRA are structured thus:

```
solenoidValve = {"solenoidValve", "Notional Helicopter Fleet",
  "I", {1998, 3, 5}, 5 10^-1, {1998, 9, 5}, 3 10^-1, {1999, 3, 5},
  1 10^-1, {1999, 9, 5}, 8 10^-2, {2000, 3, 5}, 4 10^-2, {2000, 9, 5},
  1 10^-2, {2001, 3, 5}, 7 10^-3, {2001, 9, 5}, 2 10^-3, {2002, 3, 5},
  6 10^-4, {2002, 9, 5}, 8 10^-5, {2003, 3, 5}, 0, {2003, 9, 5}}
```

{solenoidValve, Notional Helicopter Fleet, I, {1998, 3, 5}, $\frac{1}{2}$,
{1998, 9, 5}, $\frac{3}{10}$, {1999, 3, 5}, $\frac{1}{10}$, {1999, 9, 5}, $\frac{2}{25}$, {2000, 3, 5},
$\frac{1}{25}$, {2000, 9, 5}, $\frac{1}{100}$, {2001, 3, 5}, $\frac{7}{1000}$, {2001, 9, 5}, $\frac{1}{500}$,
{2002, 3, 5}, $\frac{3}{5000}$, {2002, 9, 5}, $\frac{1}{12500}$, {2003, 3, 5}, 0, {2003, 9, 5}}

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

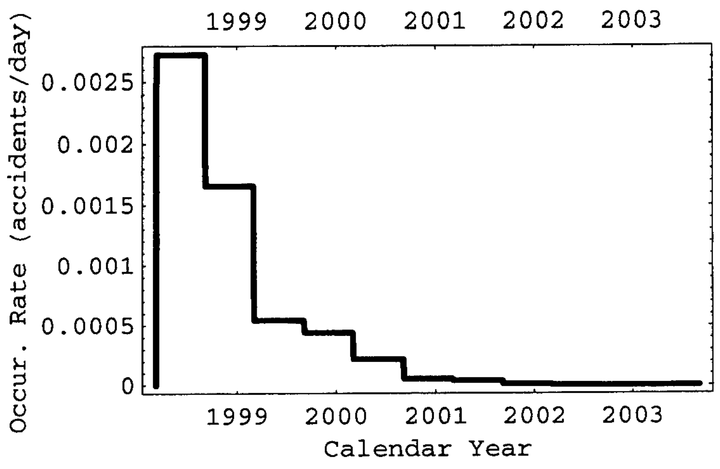
```
SSRATest[solenoidValve]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, solenoidValve];
```

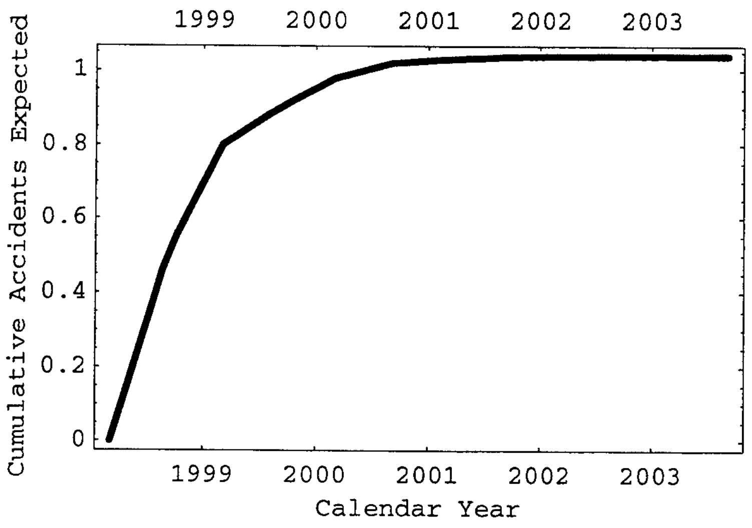
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{solenoidValve}, 1];
```



The occurrence rate decreases over time, due possibly to improved risk mitigation and/or decreasing fleet size. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{solenoidValve}, 1];
```



This plot curves downward as the occurrence rate decreases with time and levels off when the occurrence reaches zero in calendar year 2003.

- **Bolt**

The data for this SSRA are structured thus:

```
bolt = {"bolt", "Notional Helicopter Fleet", "I",
  {1998, 5, 10}, 4 10^-3, {1998, 11, 10}, 2 10^-2, {1999, 5, 10},
  6 10^-2, {1999, 11, 10}, 1 10^-1, {2000, 5, 10}, 2 10^-1,
  {2000, 11, 10}, 2 10^-1, {2001, 5, 10}, 2 10^-1, {2001, 11, 10},
  5 10^-2, {2002, 5, 10}, 9 10^-3, {2002, 11, 10}, 2 10^-3, {2003, 5, 10},
  6 10^-4, {2003, 11, 10}, 1 10^-4, {2004, 5, 10}, 0, {2004, 11, 10}}
```

$\{$bolt, Notional Helicopter Fleet, I, {1998, 5, 10}, $\frac{1}{250}$,
{1998, 11, 10}, $\frac{1}{50}$, {1999, 5, 10}, $\frac{3}{50}$, {1999, 11, 10}, $\frac{1}{10}$,
{2000, 5, 10}, $\frac{1}{5}$, {2000, 11, 10}, $\frac{1}{5}$, {2001, 5, 10}, $\frac{1}{5}$, {2001, 11, 10},
$\frac{1}{20}$, {2002, 5, 10}, $\frac{9}{1000}$, {2002, 11, 10}, $\frac{1}{500}$, {2003, 5, 10},
$\frac{3}{5000}$, {2003, 11, 10}, $\frac{1}{10000}$, {2004, 5, 10}, 0, {2004, 11, 10}$\}$

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

```
SSRATest[bolt]
```

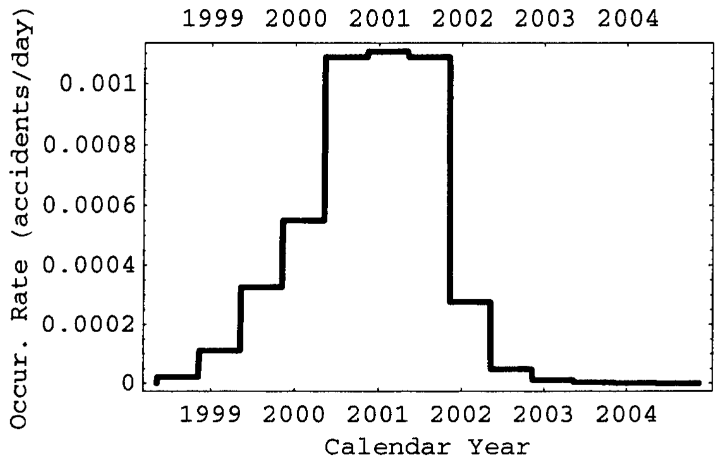
```
True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, bolt];
```

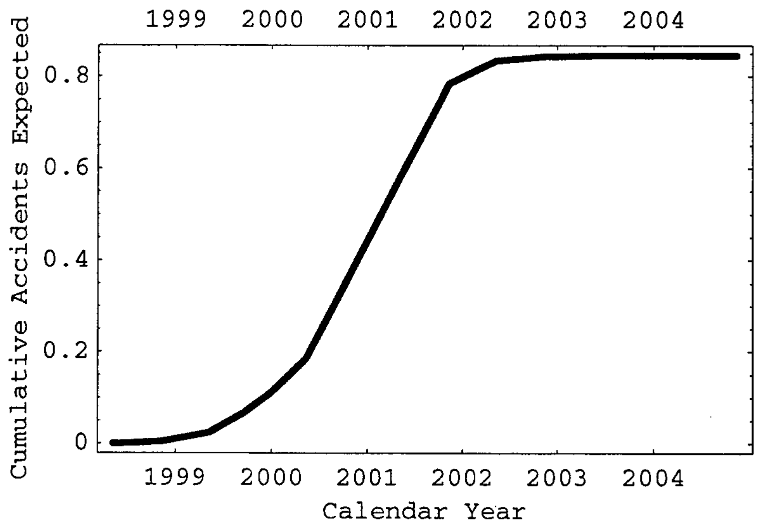
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{bolt}, 1];
```



The occurrence rate initially increases over time, due possibly to aging and/or increasing fleet size, and then decreases due to improved risk mitigation, replacement of weak components and/or decreasing fleet size. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{bolt}, 1];
```



This plot curves upward then downward as the occurrence rate increases then decreases with time, respectively. The plot levels off when the occurrence rate goes to zero.

■ Bearing

The data for this SSRA are structured thus:

```
bearing = {"bearing", "Notional Helicopter Fleet", "I",
  {1998, 7, 15}, 7 10^-4, {1999, 1, 15}, 5 10^-4, {1999, 7, 15},
  3 10^-4, {2000, 1, 15}, 1 10^-4, {2000, 7, 15}, 7 10^-5,
  {2001, 1, 15}, 5 10^-5, {2001, 7, 15}, 5 10^-5, {2002, 1, 15},
  5 10^-5, {2002, 7, 15}, 5 10^-5, {2003, 1, 15}, 5 10^-5, {2003, 7, 15},
  5 10^-5, {2004, 1, 15}, 5 10^-5, {2004, 7, 15}, 5 10^-5, {2005, 1, 15},
  5 10^-5, {2005, 7, 15}, 5 10^-5, {2006, 1, 15}, 5 10^-5, {2006, 7, 15},
  5 10^-5, {2007, 1, 15}, 5 10^-5, {2007, 7, 15}, 5 10^-5, {2008, 1, 15}}
```

$\{$bearing, Notional Helicopter Fleet, I, {1998, 7, 15}, $\frac{7}{10000}$,
{1999, 1, 15}, $\frac{1}{2000}$, {1999, 7, 15}, $\frac{3}{10000}$, {2000, 1, 15}, $\frac{1}{10000}$,
{2000, 7, 15}, $\frac{7}{100000}$, {2001, 1, 15}, $\frac{1}{20000}$, {2001, 7, 15},
$\frac{1}{20000}$, {2002, 1, 15}, $\frac{1}{20000}$, {2002, 7, 15}, $\frac{1}{20000}$,
{2003, 1, 15}, $\frac{1}{20000}$, {2003, 7, 15}, $\frac{1}{20000}$, {2004, 1, 15},
$\frac{1}{20000}$, {2004, 7, 15}, $\frac{1}{20000}$, {2005, 1, 15}, $\frac{1}{20000}$,
{2005, 7, 15}, $\frac{1}{20000}$, {2006, 1, 15}, $\frac{1}{20000}$, {2006, 7, 15},
$\frac{1}{20000}$, {2007, 1, 15}, $\frac{1}{20000}$, {2007, 7, 15}, $\frac{1}{20000}$, {2008, 1, 15}$\}$

It is not expected that this SSRA will be entirely fixed by the final date above. The SSRA prediction extends only that far. The data structure is tested thus:

```
SSRATest[bearing]
```

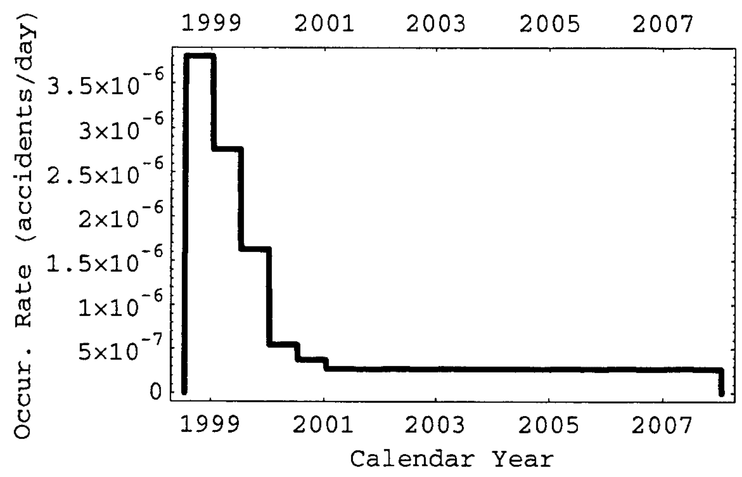
```
True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, bearing];
```

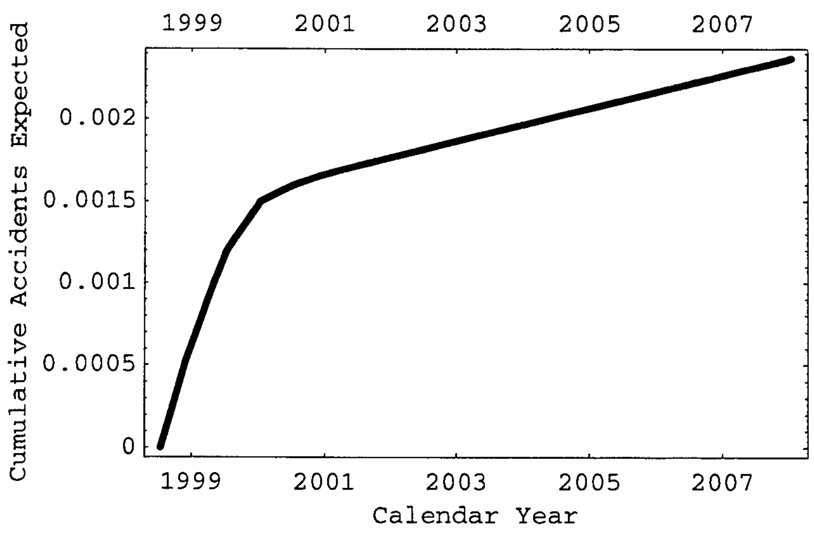
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{bearing}, 2];
```



The occurrence rate decreases over time, due possibly to improved risk mitigation and/or decreasing fleet size and then levels off in calendar year 2001. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{bearing}, 2];
```



This plot curves downward since the occurrence rate decreases with time as noted above.

- **Gear**

The data for this SSRA are structured thus:

```
gear = {"gear", "Notional Helicopter Fleet", "I",
  {1998, 9, 20}, 5 10^-3, {1999, 3, 20}, 6 10^-3, {1999, 9, 20},
  7 10^-3, {2000, 3, 20}, 8 10^-3, {2000, 9, 20}, 9 10^-3,
  {2001, 3, 20}, 1 10^-2, {2001, 9, 20}, 2 10^-2, {2002, 3, 20},
  3 10^-2, {2002, 9, 20}, 4 10^-2, {2003, 3, 20}, 5 10^-2, {2003, 9, 20},
  5 10^-2, {2004, 3, 20}, 5 10^-2, {2004, 9, 20}, 5 10^-2, {2005, 3, 20},
  5 10^-2, {2005, 9, 20}, 5 10^-2, {2006, 3, 20}, 5 10^-2, {2006, 9, 20},
  5 10^-2, {2007, 3, 20}, 5 10^-2, {2007, 9, 20}, 5 10^-2, {2008, 3, 20}}
```

```
- General::spell1 : Possible spelling error:
    new symbol name "gear" is similar to existing symbol "Gear".
```

$$\{\text{gear, Notional Helicopter Fleet, I}, \{1998, 9, 20\}, \frac{1}{200}, \{1999, 3, 20\}, \frac{3}{500}, \{1999, 9, 20\}, \frac{7}{1000}, \{2000, 3, 20\}, \frac{1}{125}, \{2000, 9, 20\}, \frac{9}{1000}, \{2001, 3, 20\}, \frac{1}{100}, \{2001, 9, 20\}, \frac{1}{50}, \{2002, 3, 20\}, \frac{3}{100}, \{2002, 9, 20\}, \frac{1}{25}, \{2003, 3, 20\}, \frac{1}{20}, \{2003, 9, 20\}, \frac{1}{20}, \{2004, 3, 20\}, \frac{1}{20}, \{2004, 9, 20\}, \frac{1}{20}, \{2005, 3, 20\}, \frac{1}{20}, \{2005, 9, 20\}, \frac{1}{20}, \{2006, 3, 20\}, \frac{1}{20}, \{2006, 9, 20\}, \frac{1}{20}, \{2007, 3, 20\}, \frac{1}{20}, \{2007, 9, 20\}, \frac{1}{20}, \{2008, 3, 20\}\}$$

It is not expected that this SSRA will be entirely fixed by the final date above. The SSRA prediction extends only that far.

It would be wise to consider why *Mathematica* generated the error message above. Additional information on the symbol *Gear* is obtained thus:

```
? Gear
```

```
Gear is a choice for the option Method of NDSolve. Method -> Gear is a
  backwards differentiation method specifically for stiff equations.
```

The existing symbol *Gear* pertains to the built-in function `NDSolve` and was not defined herein due to a errant keystroke.

The data structure is tested thus:

```
SSRATest[gear]
```

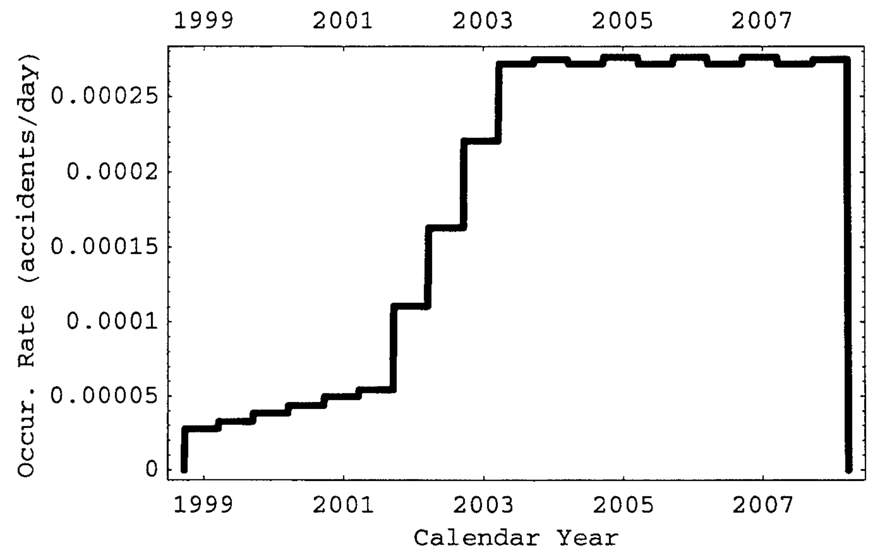
```
True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, gear];
```

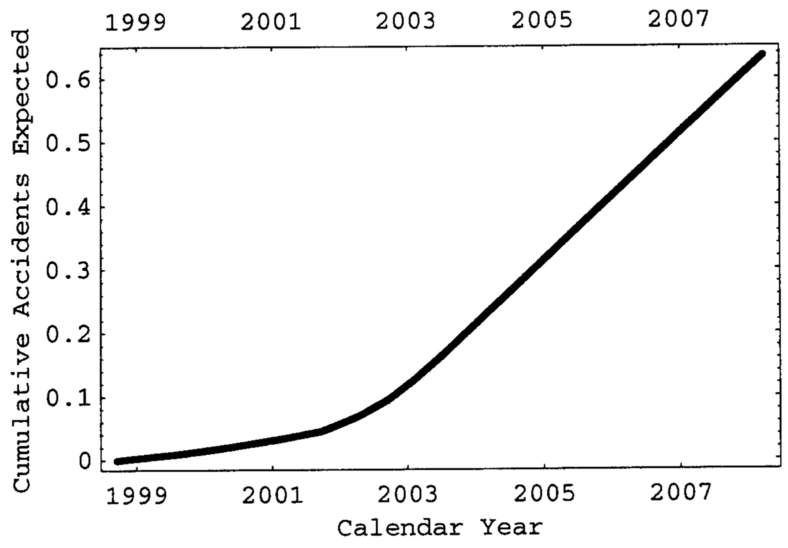
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{gear}, 2];
```



The occurrence rate increases over time, due possibly to aging and/or increasing fleet size and then levels off in calendar year 2003. The slight perturbations after the start of 2003 are due to the months being of different length year. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{gear}, 2];
```



This plot curves upward and then straightens out as the occurrence rate increases and then levels off as noted above.

- **Solenoid**

The data for this SSRA are structured thus:

solenoid = {"solenoid", "Notional Helicopter Fleet", "II", {1998, 11, 25}, 8 10^{-2}, {1999, 5, 25}, 3 10^{-2}, {1999, 11, 25}, 9 10^{-3}, {2000, 5, 25}, 7 10^{-3}, {2000, 11, 25}, 3 10^{-3}, {2001, 5, 25}, 1 10^{-3}, {2001, 11, 25}, 8 10^{-4}, {2002, 5, 25}, 5 10^{-4}, {2002, 11, 25}, 2 10^{-4}, {2003, 5, 25}, 9 10^{-5}, {2003, 11, 25}, 8 10^{-5}, {2004, 5, 25}, 6 10^{-5}, {2004, 11, 25}, 1 10^{-5}, {2005, 5, 25}, 0, {2005, 11, 25}}

{solenoid, Notional Helicopter Fleet, II, {1998, 11, 25}, $\frac{2}{25}$, {1999, 5, 25}, $\frac{3}{100}$, {1999, 11, 25}, $\frac{9}{1000}$, {2000, 5, 25}, $\frac{7}{1000}$, {2000, 11, 25}, $\frac{3}{1000}$, {2001, 5, 25}, $\frac{1}{1000}$, {2001, 11, 25}, $\frac{1}{1250}$, {2002, 5, 25}, $\frac{1}{2000}$, {2002, 11, 25}, $\frac{1}{5000}$, {2003, 5, 25}, $\frac{9}{100000}$, {2003, 11, 25}, $\frac{1}{12500}$, {2004, 5, 25}, $\frac{3}{50000}$, {2004, 11, 25}, $\frac{1}{100000}$, {2005, 5, 25}, 0, {2005, 11, 25}}

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

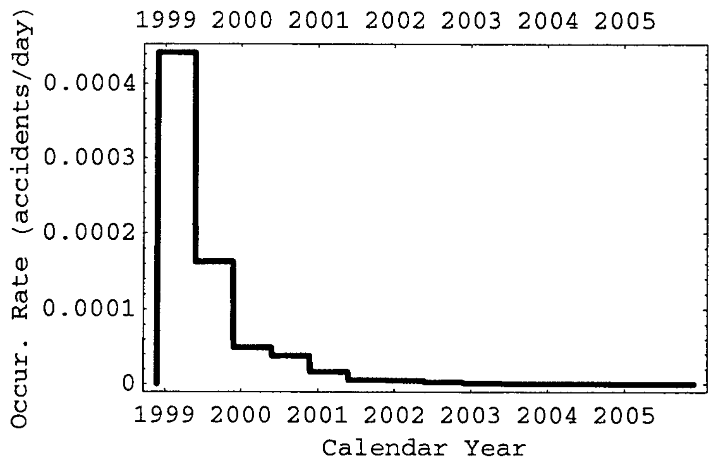
```
SSRATest[solenoid]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, solenoid];
```

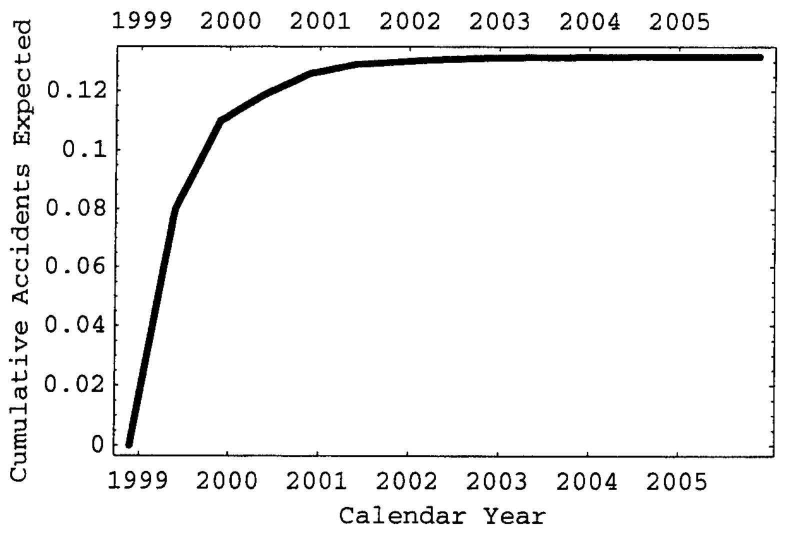
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{solenoid}, 1];
```



The occurrence rate decreases over time, due possibly to improved risk mitigation and/or decreasing fleet size, and drops to zero in calendar year 2005. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{solenoid}, 1];
```



This plot curves downward since the occurrence rate decreases with time as noted above and levels off when the occurrence rate reaches zero.

- **Mount**

The data for this SSRA are structured thus:

```
mount = {"mount", "Notional Helicopter Fleet", "II", {1999, 2, 1},
  6 10^-4, {1999, 8, 1}, 1 10^-3, {2000, 2, 1}, 4 10^-3, {2000, 8, 1},
  9 10^-3, {2001, 2, 1}, 2 10^-2, {2001, 8, 1}, 3 10^-2, {2002, 2, 1},
  3 10^-2, {2002, 8, 1}, 3 10^-2, {2003, 2, 1}, 1 10^-2, {2003, 8, 1},
  6 10^-3, {2004, 2, 1}, 3 10^-3, {2004, 8, 1}, 9 10^-4, {2005, 2, 1},
  5 10^-4, {2005, 8, 1}, 1 10^-4, {2006, 2, 1}, 0, {2006, 8, 1}}
```

$\{$mount, Notional Helicopter Fleet, II, {1999, 2, 1}, $\frac{3}{5000}$, {1999, 8, 1}, $\frac{1}{1000}$, {2000, 2, 1}, $\frac{1}{250}$, {2000, 8, 1}, $\frac{9}{1000}$, {2001, 2, 1}, $\frac{1}{50}$, {2001, 8, 1}, $\frac{3}{100}$, {2002, 2, 1}, $\frac{3}{100}$, {2002, 8, 1}, $\frac{3}{100}$, {2003, 2, 1}, $\frac{1}{100}$, {2003, 8, 1}, $\frac{3}{500}$, {2004, 2, 1}, $\frac{3}{1000}$, {2004, 8, 1}, $\frac{9}{10000}$, {2005, 2, 1}, $\frac{1}{2000}$, {2005, 8, 1}, $\frac{1}{10000}$, {2006, 2, 1}, 0, {2006, 8, 1}$\}$

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

```
SSRATest[mount]
```

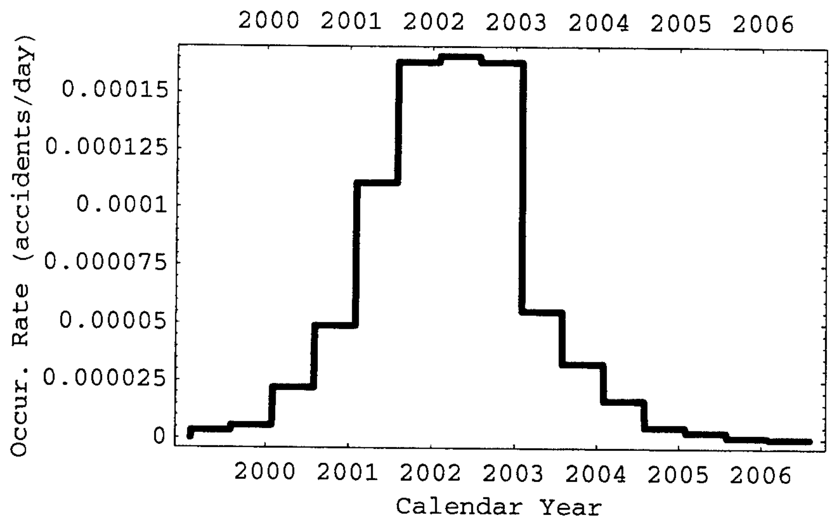
```
True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, mount];
```

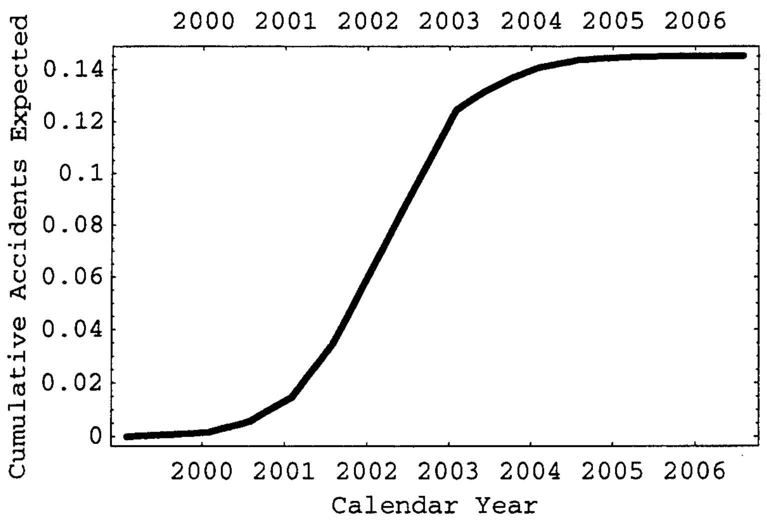
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{mount}, 1];
```



The occurrence rate initially increases over time, due possibly to aging and/or increasing fleet size, and then decreases due to improved risk mitigation, replacement of weak components and/or decreasing fleet size. The occurrence rate drops to zero in calendar year 2006. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{mount}, 1];
```



This plot curves upward then downward as the occurrence rate increases then decreases with time as noted above. The plot levels off when the occurrence rate drops to zero.

▪ Plate

The data for this SSRA are structured thus:

```
plate = {"plate", "Notional Helicopter Fleet", "II",
  {1999, 4, 5}, 2 10^-1, {1999, 10, 5}, 7 10^-2, {2000, 4, 5}, 4 10^-2,
  {2000, 10, 5}, 1 10^-2, {2001, 4, 5}, 8 10^-3, {2001, 10, 5},
  5 10^-3, {2002, 4, 5}, 2 10^-3, {2002, 10, 5}, 9 10^-4, {2003, 4, 5},
  6 10^-4, {2003, 10, 5}, 3 10^-4, {2004, 4, 5}, 1 10^-4, {2004, 10, 5},
  7 10^-5, {2005, 4, 5}, 3 10^-5, {2005, 10, 5}, 0, {2006, 4, 5}}
```

{plate, Notional Helicopter Fleet, II, {1999, 4, 5}, $\frac{1}{5}$,
{1999, 10, 5}, $\frac{7}{100}$, {2000, 4, 5}, $\frac{1}{25}$, {2000, 10, 5}, $\frac{1}{100}$,
{2001, 4, 5}, $\frac{1}{125}$, {2001, 10, 5}, $\frac{1}{200}$, {2002, 4, 5}, $\frac{1}{500}$,
{2002, 10, 5}, $\frac{9}{10000}$, {2003, 4, 5}, $\frac{3}{5000}$, {2003, 10, 5},
$\frac{3}{10000}$, {2004, 4, 5}, $\frac{1}{10000}$, {2004, 10, 5}, $\frac{7}{100000}$,
{2005, 4, 5}, $\frac{3}{100000}$, {2005, 10, 5}, 0, {2006, 4, 5}}

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

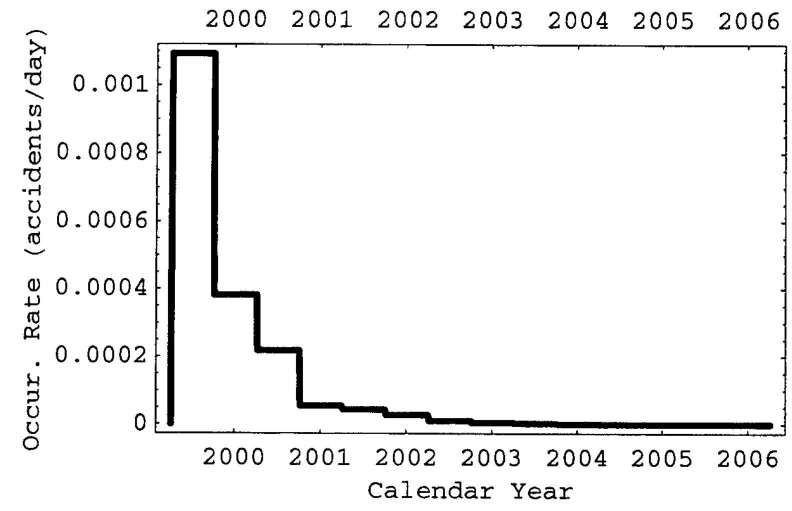
```
SSRATest[plate]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, plate];
```

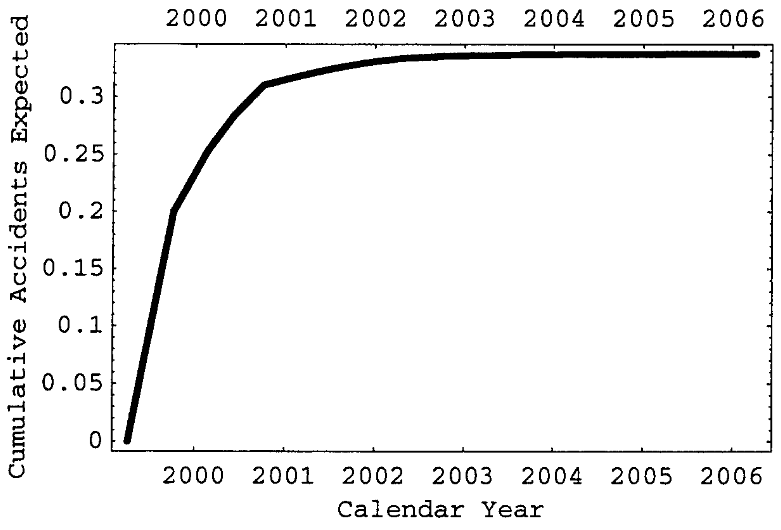
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{plate}, 1];
```



The occurrence rate decreases over time, due possibly to improved risk mitigation and/or decreasing fleet size. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{plate}, 1];
```



This plot curves downward as the occurrence rate decreases with time and levels off when the occurrence rate reaches zero.

Rotor Blade

The data for this SSRA are structured thus:

```
rotorBlade =
 {"rotorBlade", "Notional Helicopter Fleet", "II", {1999, 6, 10},
  1 10^-3, {1999, 12, 10}, 5 10^-3, {2000, 6, 10}, 7 10^-3, {2000, 12, 10},
  1 10^-2, {2001, 6, 10}, 3 10^-2, {2001, 12, 10}, 5 10^-2, {2002, 6, 10},
  5 10^-2, {2002, 12, 10}, 5 10^-2, {2003, 6, 10}, 5 10^-2, {2003, 12, 10},
  5 10^-2, {2004, 6, 10}, 5 10^-2, {2004, 12, 10}, 5 10^-2, {2005, 6, 10},
  5 10^-2, {2005, 12, 10}, 5 10^-2, {2006, 6, 10}, 5 10^-2, {2006, 12, 10},
  5 10^-2, {2007, 6, 10}, 5 10^-2, {2007, 12, 10}, 5 10^-2, {2008, 6, 10}}
```

$\{$rotorBlade, Notional Helicopter Fleet, II, {1999, 6, 10},
$\frac{1}{1000}$, {1999, 12, 10}, $\frac{1}{200}$, {2000, 6, 10}, $\frac{7}{1000}$, {2000, 12, 10},
$\frac{1}{100}$, {2001, 6, 10}, $\frac{3}{100}$, {2001, 12, 10}, $\frac{1}{20}$, {2002, 6, 10},
$\frac{1}{20}$, {2002, 12, 10}, $\frac{1}{20}$, {2003, 6, 10}, $\frac{1}{20}$, {2003, 12, 10},
$\frac{1}{20}$, {2004, 6, 10}, $\frac{1}{20}$, {2004, 12, 10}, $\frac{1}{20}$, {2005, 6, 10},
$\frac{1}{20}$, {2005, 12, 10}, $\frac{1}{20}$, {2006, 6, 10}, $\frac{1}{20}$, {2006, 12, 10},
$\frac{1}{20}$, {2007, 6, 10}, $\frac{1}{20}$, {2007, 12, 10}, $\frac{1}{20}$, {2008, 6, 10}$\}$

It is expected that this SSRA will not be entirely fixed by the final date above. The SSRA prediction extends only that far. The data structure is tested thus:

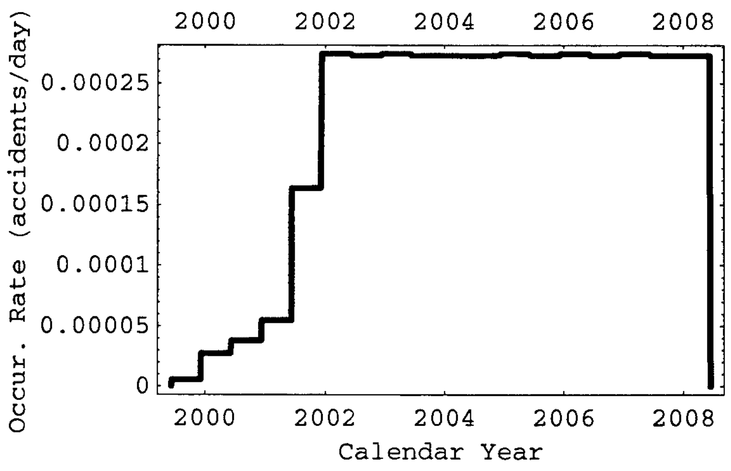
```
SSRATest[rotorBlade]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, rotorBlade];
```

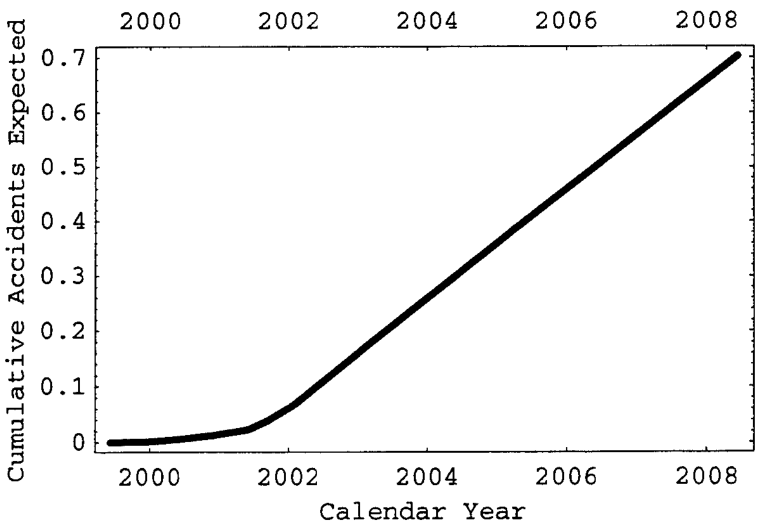
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{rotorBlade}, 2];
```



The occurrence rate increases over time, due possibly to aging and/or increasing fleet size and then levels off. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{rotorBlade}, 2];
```



This plot curves upward while the occurrence rate increases with time and then straightens as the occurrence rate levels off.

- **Shutoff Valve**

The data for this SSRA are structured thus:

```
shutoffValve = {"shutoffValve", "Notional Helicopter Fleet",
  "II", {1999, 8, 15}, 5 10^-3, {2000, 2, 15}, 3 10^-3, {2000, 8, 15},
  1 10^-3, {2001, 2, 15}, 8 10^-4, {2001, 8, 15}, 6 10^-4,
  {2002, 2, 15}, 4 10^-4, {2002, 8, 15}, 2 10^-4, {2003, 2, 15},
  9 10^-5, {2003, 8, 15}, 6 10^-5, {2004, 2, 15}, 3 10^-5, {2004, 8, 15},
  1 10^-5, {2005, 2, 15}, 8 10^-6, {2005, 8, 15}, 5 10^-6, {2006, 2, 15},
  3 10^-6, {2006, 8, 15}, 1 10^-6, {2007, 2, 15}, 0, {2007, 8, 15}}
```

{shutoffValve, Notional Helicopter Fleet, II, {1999, 8, 15}, $\frac{1}{200}$, {2000, 2, 15}, $\frac{3}{1000}$, {2000, 8, 15}, $\frac{1}{1000}$, {2001, 2, 15}, $\frac{1}{1250}$, {2001, 8, 15}, $\frac{3}{5000}$, {2002, 2, 15}, $\frac{1}{2500}$, {2002, 8, 15}, $\frac{1}{5000}$, {2003, 2, 15}, $\frac{9}{100000}$, {2003, 8, 15}, $\frac{3}{50000}$, {2004, 2, 15}, $\frac{3}{100000}$, {2004, 8, 15}, $\frac{1}{100000}$, {2005, 2, 15}, $\frac{1}{125000}$, {2005, 8, 15}, $\frac{1}{200000}$, {2006, 2, 15}, $\frac{3}{1000000}$, {2006, 8, 15}, $\frac{1}{1000000}$, {2007, 2, 15}, 0, {2007, 8, 15}}

It is expected that this SSRA will be entirely fixed by the final date above. The data structure is tested thus:

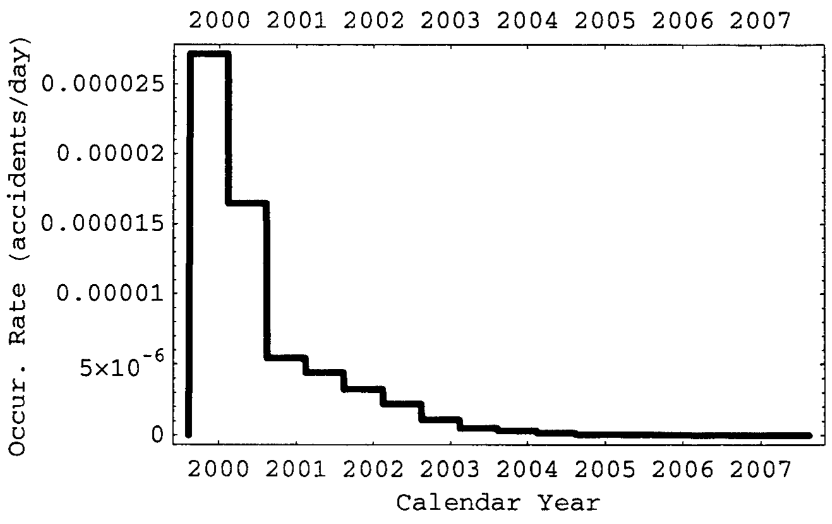
```
SSRATest[shutoffValve]

True
```

The SSRA is correctly structured and is now appended to *HelicopterSSRAs*:

```
AppendTo[HelicopterSSRAs, shutoffValve];
```

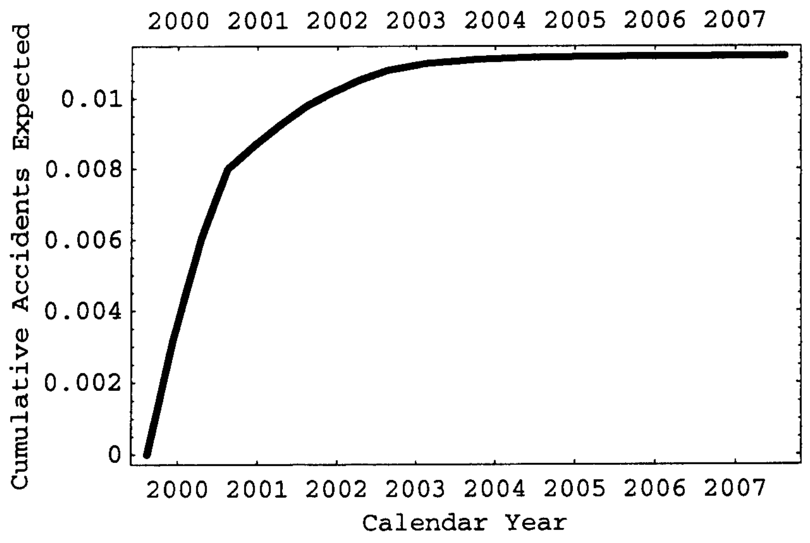
The occurrence-rate plot for this SSRA is:

```
OccurrenceRatePlot[{shutoffValve}, 1];
```



The occurrence rate decreases over time, due possibly to improved risk mitigation and/or decreasing fleet size. A plot of the number of accidents expected to accumulate over the entire life of this SSRA is:

```
ExpectedAccidentsPlot[{shutoffValve}, 1];
```



This plot curves downward as the occurrence rate decreases with time and then levels off when the occurrence rate drops to zero in calendar year 2007.

■ Calendar Time Interval for Analysis

Predictions for three of the notional SSRAs expire before they are expected to be fixed:

▪ bearing	{2008, 1, 15}
▪ gear	{2008, 3, 20}
▪ rotor blade	{2008, 6, 10}

One should take note of the earliest date since analysis beyond this date will be misleading. The expiration of SSRA predictions may be interpreted as the fixing of SSRAs.

Preparation and Input of Flight-Hour Data

In order to relate expected-accident quantities and probabilities to the helicopter-fleet flight-hour profile in various EAA model functions, data is needed concerning the flight hours to be flown by the notional helicopter fleet as a function of calendar time. The flight-hours data for the notional helicopter fleet is as follows:

```
HelicopterHours = {{1998, 10, 1}, 60000, {1999, 10, 1}, 65000,
   {2000, 10, 1}, 70000, {2001, 10, 1}, 75000, {2002, 10, 1},
   80000, {2003, 10, 1}, 85000, {2004, 10, 1}, 76176,
   {2005, 10, 1}, 74832, {2006, 10, 1}, 73296, {2007, 10, 1}, 71952,
   {2008, 10, 1}, 70608, {2009, 10, 1}, 69264, {2010, 10, 1}, 67728,
   {2011, 10, 1}, 66384, {2012, 10, 1}, 65040, {2013, 10, 1}, 63696,
   {2014, 10, 1}, 62160, {2015, 10, 1}, 60816, {2016, 10, 1}, 59472,
   {2017, 10, 1}, 58128, {2018, 10, 1}, 56592, {2019, 10, 1}, 55248,
   {2020, 10, 1}, 53904, {2021, 10, 1}, 52368, {2022, 10, 1}};
```

It can be seen from the example above that the flight-hours data for the helicopter fleet consists of the following information:

- calendar start date for the first flight-hours prediction (a list of integers formatted as {yyyy, m, d})
- prediction of the number of fleet flight hours between above date and next date
- date prediction above ends and next prediction, if there is one, begins (formatted as date above)

The last two elements above are repeated as necessary. The flight-hours data must be structured in a specific manner in order for the new add-on functions that implement the EAA model to work. The structuring of the flight-hours data is presented after the SSRA data.The new add-on function `FlightHoursTest` should be used to test the structure of this data. The usage message for this function is:

```
?FlightHoursTest

FlightHoursTest[FHdata] tests a flight-
  hours list to ensure it's structured correctly.
```

The flight-hours data, respectively, are tested thus:

```
FlightHoursTest[HelicopterHours]

True
```

The flight-hour data are structured correctly.

Sort and Save Data to a File

It will be helpful during subsequent analysis to sort the SSRAs into alphabetical order. The lists of SSRAs are sorted alphabetically as follows:

```
HelicopterSSRAs = Sort[HelicopterSSRAs, OrderedQ[{#1[[1]], #2[[1]]}] &];
```

The quantity of SSRAs to be analyzed in the next chapter are:

```
Length[HelicopterSSRAs]

10
```

Now the lists of SSRAs and flight hours for the notional fleet will be saved to a file. First we will assemble, in a platform-independent way, a name for a file where the data will be stored:

```
datafilename =
 ToFileName[{$TopDirectory, "AddOns", "ExtraPackages", "AviationSafety",
   "ExpAccAggModel", "MethodologyTR"}, "HelicopterDataFile"]

C:\Program Files\Wolfram Research\Mathematica\4.0\AddOns\ExtraPackages\
  AviationSafety\ExpAccAggModel\MethodologyTR\HelicopterDataFile
```

Any existing versions of this file will first be deleted:

```
DeleteFile[datafilename]
```

The expected-accident data is now saved to the file:

```
Save[datafilename, {HelicopterSSRAs, HelicopterHours}]
```

Summary

In this chapter, the notional SSRA data were examined and formatted so that the EAA model can be applied in the next chapter. This chapter includes a subsection on each SSRA that contains details including graphs and observations on the time-dependent behavior of each SSRA. The SSRA data structure for the helicopter fleet was assembled and saved to a file for analysis and aggregation in the next chapter. The flight-hours data was structured and saved to a file as well.

Chapter 4

Aggregation and Analysis of the Helicopter Fleet SSRAs

Introduction

This chapter contains the aggregation and analysis of the notional helicopter fleet SSRAs prepared in Chapter 3. Analysis sections of this chapter are dedicated to the following categories of SSRAs:

- severity-level I SSRAs
- severity-level II SSRAs
- severity-level I and II SSRAs

The section on severity-level I SSRAs is the longest since many of the analysis functions for the Expected Accident Aggregation (EAA) model are introduced in this report for the first time.

Whenever SSRAs from more than one severity level are combined, this is done without weighting. One could also view this as equal weighting.

Several pre-analysis tasks must be performed before turning to the analysis of the SSRAs.

Analysis Preliminaries

A few preliminary steps, including loading of the EAA functions and retrieving the SSRA data, must be taken before the analysis can begin.

■ Loading Additional Functions

The add-on functions for the EAA model are defined by the new *Mathematica* package found at Appendix A. The package is loaded with the built-in function `Needs`:

```
Needs["AviationSafety`ExpectedAccidentAggregationModel`"]
```

The usage message for this package is:

```
? ExpectedAccidentAggregationModel
```

```
ExpectedAccidentAggregationModel.m (version 1.0.1) is a package that
  contains functions for the Expected Accident Aggregation model.
```

We also need to load the standard *Mathematica* add-on package `Statistics`DiscreteDistribution`` in order to calculate and plot accident probabilities in each of the remaining sections of this chapter.

```
Needs["Statistics`DiscreteDistributions`"]
```

The built-in function `ListPlot` will also be used for probability plots. It is helpful to modify the default settings for selected plotting options so they won't have to be specified each time `ListPlot` is used in this chapter. The desired, non-default values for the options of `ListPlot` are set thus:

```
SetOptions[ListPlot, Frame -> True, Axes -> False,
GridLines→{Automatic, Automatic}, PlotStyle→PointSize[.02],
FrameTicks→{Automatic, Automatic}, PlotRange→Automatic];
```

It will also be helpful to use color on plots where multiple SSRAs are plotted. Rather than specify the color options each time, the colors are specified once here and assigned as the value of the symbol *colors*. Whenever needed throughout this chapter, these colors can be invoked by using this symbol.

```
colors = {Hue[1], Hue[.1], Hue[.9],
   Hue[.2], Hue[.8], Hue[.3], Hue[.6], Hue[.4], Hue[.5]};
```

■ Retrieve Data on SSRAs and Flight Hours from File

In Chapter 3, the expected-accident data for the notional helicopter fleet were saved to a file for subsequent aggregation and analysis. These data must now be retrieved. First we will assemble, in a platform-independent way (using the built-in function `ToFileName`), the name of the file where the data were stored:

```
datafilename =
 ToFileName[{$TopDirectory, "AddOns", "ExtraPackages", "AviationSafety",
   "ExpAccAggModel", "MethodologyTR"}, "HelicopterDataFile"]
```

```
C:\Program Files\Wolfram Research\Mathematica\4.0\AddOns\ExtraPackages\
  AviationSafety\ExpAccAggModel\MethodologyTR\HelicopterDataFile
```

The SSRA expected-accident and flight-hour data are now obtained with the built-in function `Get`:

```
Get[datafilename];
```

The names of the lists of SSRAs and flight hours obtained are:

```
?Helicopter*
```

```
HelicopterHours HelicopterSSRAs
```

Analysis Dates

Each of the add-on functions for the EAA model will accept any date(s) the analyst chooses. It will be helpful to define a current date as the value of a symbol so that we need only specify it once. We will assign 3 Jan 2002 as the value of the symbol *today*:

```
today = {2002, 1, 3};
```

All dates must be in this format.

The end of 2007 will be used as the ending date for in this chapter since, in 2008, some of the notional SSRA predictions formulated in the previous chapter begin to expire. These SSRAs are not expected to be completely cured by then, it's just that their predictions were only taken so many years into the future.

The Most Recent SSRA

One aspect of the analysis is to examine the impact of the most recent SSRA. The new add-on function `LastSSRA` will extract the SSRA that began last:

```
?LastSSRA
```

```
LastSSRA[SSRAlist] selects from SSRAlist the SSRA that began last.
```

The SSRA that began last is:

```
LastSSRA[HelicopterSSRAs]
```

```
{shutoffValve, Notional Helicopter Fleet, II, {1999, 8, 15},
 1/200, {2000, 2, 15}, 3/1000, {2000, 8, 15}, 1/1000, {2001, 2, 15},
 1/1250, {2001, 8, 15}, 3/5000, {2002, 2, 15}, 1/2500, {2002, 8, 15},
 1/5000, {2003, 2, 15}, 9/100000, {2003, 8, 15}, 3/50000,
 {2004, 2, 15}, 3/100000, {2004, 8, 15}, 1/100000, {2005, 2, 15},
 1/125000, {2005, 8, 15}, 1/200000, {2006, 2, 15}, 3/1000000,
 {2006, 8, 15}, 1/1000000, {2007, 2, 15}, 0, {2007, 8, 15}}
```

Since this is a severity-level II SSRA, its impact will be considered in that section.

Analysis of Severity-Level I SSRAs

In this section, the severity-level I SSRAs are analyzed. Since the most recent SSRA does not fall into this category, it will not be addressed here. The impact of the most recent SSRA will, however, be addressed in the section on severity-level II SSRAs. Finally, the impact of immediately fixing the worst SSRAs is considered.

Many of the add-on functions that implement the EAA Model will be introduced in this section.

■ Aggregation and Analysis

First, the severity-level I SSRAs must be extracted from the complete list of SSRAs for the notional helicopter fleet. The new add-on function `SSRASelect` can be used for this purpose:

```
? SSRASelect
```

```
SSRASelect[SSRAlistAll, sevlev] selects
  all SSRAs in SSRAlistAll with severity level sevlev.
```

The severity-level I SSRAs are extracted and assigned as the value of the symbol *ssralist* thus:

```
ssralist = SSRASelect[HelicopterSSRAs, "I"];
```

A formatted table of occurrence rates for the SSRAs in *ssralist* might be of interest. The new add-on function `OccurrenceRateTable` may be used for this purpose:

```
?OccurrenceRateTable
```

```
OccurrenceRateTable[SSRAlist, date, opts] generates
  a table of occurrence rates in accidents per day on the
  specified date for all SSRAs in SSRAlist. OccurrenceRateTable[
  SSRAlist, nameList, date, opts] generates a table of
  occurrence rates for only SSRAs in nameList. opts is
  an optional argument for specifying TableForm options.
```

A table of *today*'s occurrence rates for the SSRAs in *ssralist*, in decreasing order, is generated as follows:

```
OccurrenceRateTable[ssralist, today] // N
```

```
SSRA              OCCURRENCE RATE (ACCIDENTS/DAY)      PERCENT
bolt              0.000276243                          68.447
gear              0.000110497                          27.3788
solenoidValve     0.0000110497                         2.73788
clutch            5.52486 x 10^-6                      1.36894
bearing           2.71739 x 10^-7                      0.067331
SUM               0.000403587                          100.
```

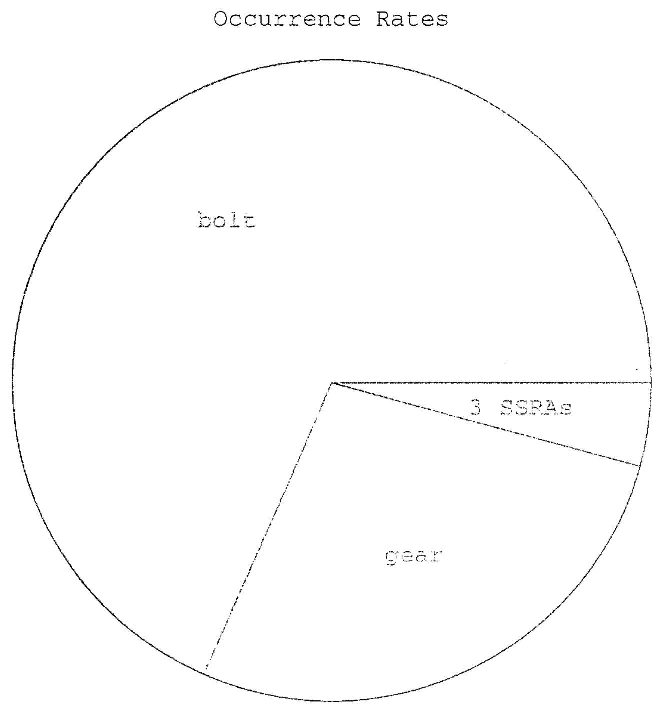
Upon review of the percentages above, it is clear that the first two SSRAs are quite dominant. A pie chart of these occurrence rates might also be of helpful. The new add-on function `OccurrenceRatePie` will accomplish this:

```
?OccurrenceRatePie
```

```
OccurrenceRatePie[SSRAlist, date, topSSRAs, opts] generates
  a pie chart of occurrence rates in accidents per day on the
  specified date for the SSRAs in SSRAlist. Only the largest
  SSRAs, as specified by topSSRAs, are given their own wedges.
  The remaining SSRAs are placed in a single wedge.  opts
  is an optional argument for specifying PieChart options.
```

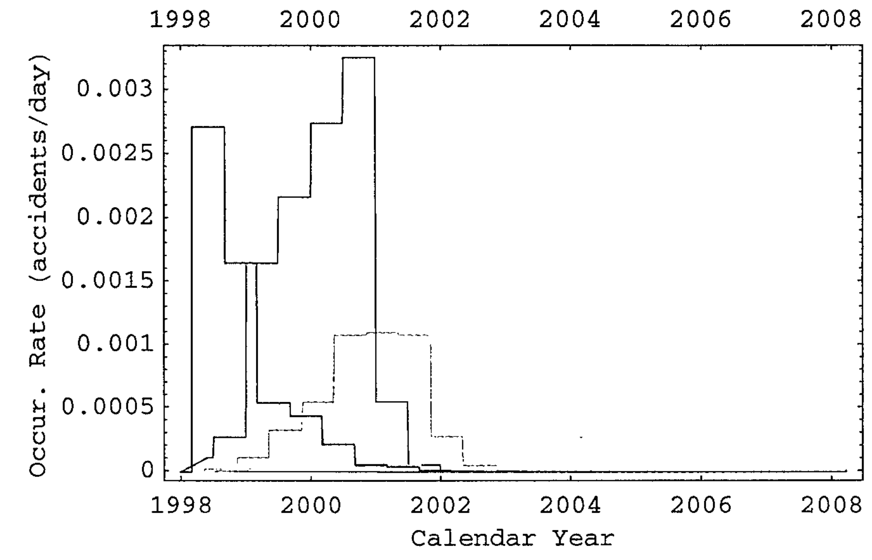
A pie chart of *today*'s occurrence rates, with the top two SSRAs given their own wedges, is generated thus:

```
OccurrenceRatePie[ssralist, today, 2, PieStyle → colors];
```



The table and pie chart above represent only a snapshot in time. It may be helpful to plot the occurrence rates over time. The new add-on function `OccurrenceRatePlot`, which was introduced in the previous chapter, may be used for this purpose. The occurrence rates of all the SSRAs in *ssralist*, from a day before the earliest begins until a day after the last ends, are plotted as follows:

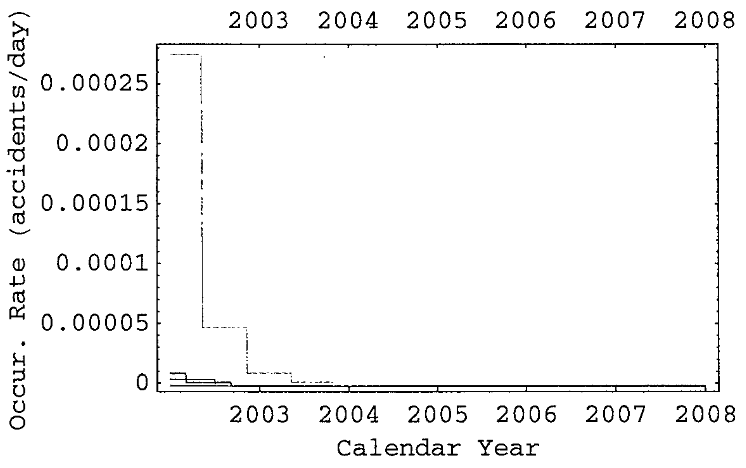
```
OccurrenceRatePlot[ssralist, 2, PlotStyle → colors];
```



Inspection of the plot above reveals that the "solenoidValve" SSRA was quite dominant in 1998 but declined and was surpassed by the "clutch" SSRA in 1999. The "bolt" SSRA surpassed the "clutch" SSRA in 2001. The "gear" SSRA is expected to dominate in 2003.

While the plot above is useful for historical purposes, it would also be helpful to examine the trends from *today* forward more closely. The occurrence rates of all the SSRAs in *ssralist* from *today* until the end of calendar year 2007 are plotted as follows:

```
OccurrenceRatePlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → colors];
```



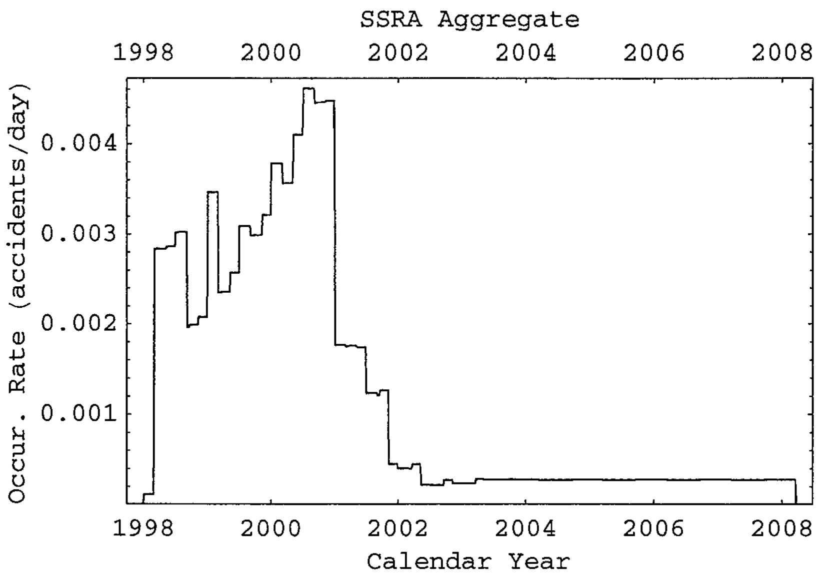
This presents a much simpler picture. The occurrence rate that is initially uppermost represents the "bolt" SSRA. This curve declines rapidly during the next couple of years due presumably to the replacement of weak components, improved risk mitigation and/or declining helicopter fleet size. The occurrence rate of the "gear" SSRA increases rapidly and will soon be quite dominant. It should now be the focus of additional risk-mitigation efforts.

It would be useful to plot the aggregate of these occurrence rates. The new add-on function `AggregateOccurrenceRatePlot` was developed for this purpose:

```
? AggregateOccurrenceRatePlot

AggregateOccurrenceRatePlot[SSRAlist, yearInc, opts] plots the
  aggregated occurrence rate (accidents per day) of all SSRAs in
  SSRAlist from a day before the earliest begins to a day after
  the last ends. yearInc specifies the calendar-year increments
  to be used on the plot. AggregateOccurrenceRatePlot[SSRAlist,
  {startDate, endDate}, yearInc, opts] plots the aggregated
  occurrence rate of all SSRAs from startDate to endDate.
  AggregateOccurrenceRatePlot[SSRAlist, nameList, yearInc,
  opts] plots the aggregated occurrence rate of only SSRAs in
  nameList from a day before the earliest begins to a day after
  the last ends. AggregateOccurrenceRatePlot[SSRAlist, nameList,
  {startDate, endDate}, yearInc, opts] plots the aggregated
  occurrence rate of only SSRAs in nameList from startDate to
  endDate. opts is an optional argument for specifying Plot options.
```

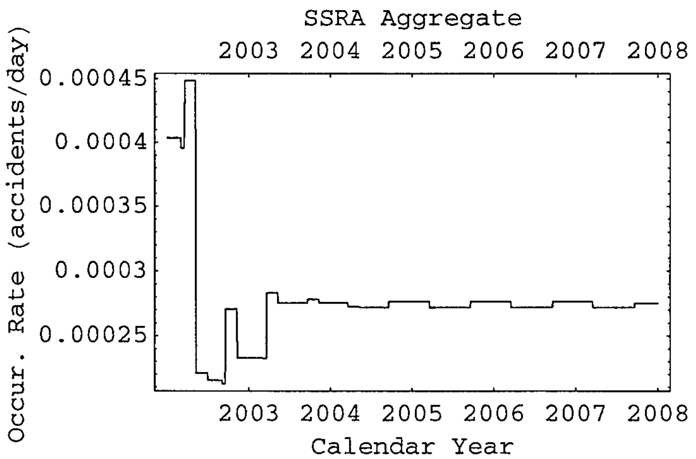
A plot of the trend in aggregate occurrence rate from the beginning of the first SSRA until the end of the last is:

```
AggregateOccurrenceRatePlot[ssralist, 2, PlotStyle → RGBColor[0, 0, 1]];
```



A plot such as this can be used to help one visualize the historical trend in SSRA risk predictions. The plot above can show that predicted SSRA risks are now declining sharply. A plot of the trend in aggregate occurrence rate from *today* until the end of 2007 is:

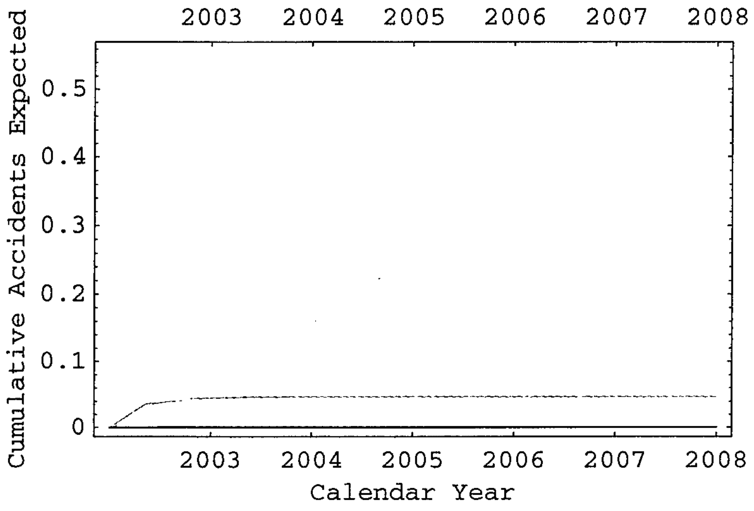
```
AggregateOccurrenceRatePlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → RGBColor[0, 0, 1]];
```



The aggregate occurrence rate will decline a bit more and then level off. This follows the pattern of the dominant SSRAs, as expected.

It would be useful to plot the expected number of accidents for each of the SSRAs. The new add-on function ExpectedAccidentsPlot, which was introduced in the previous chapter, will generate the desired plot. The cumulative expected-accident curves for each of the SSRAs in *ssralist*, from *today* until the end of 2007, are plotted thus:

```
ExpectedAccidentsPlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → colors];
```



The cumulative expected-accident curve is the sum of the area under the occurrence-rate curve. The cumulative expected-accident curve for the "bolt" SSRA curves downward because its occurrence rate decreases over time as noted above. The curve for the "gear" SSRA curves initially upward and then straightens since its occurrence rate decreases and then levels off.

The new add-on function ExpectedAccidentsTable will generate an expected-accidents table for these SSRAs:

```
? ExpectedAccidentsTable

ExpectedAccidentsTable[SSRAlist, startDate, year, opts]
  generates a table of n-year expected accidents for all SSRAs
  in SSRAlist from startDate forward. ExpectedAccidentsTable[
  SSRAlist, nameList, startDate, year, opts] generates a table
  of n-year expected accidents for only SSRAs in nameList
  from startDate forward. ExpectedAccidentsTable[SSRAlist,
  startDate, yearList, opts] generates a table for all SSRAs
  with multiple expected-accident columns where the years
  are specified by yearList. ExpectedAccidentsTable[SSRAlist,
  nameList, startDate, yearList, opts] generates a table with
  multiple expected-accident columns for only SSRAs in nameList.
  opts is an optional argument for specifying TableForm options.
```

A table of expected-accident values from *today* forward three years, for all the SSRAs in *ssralist*, is generated as follows:

```
ExpectedAccidentsTable[ssralist, today, 3] // N

SSRA               EXP. ACC. 3 YEAR(s)      PERCENT
gear               0.257403                 83.8914
bolt               0.0467829                15.2472
solenoidValve      0.00135403               0.441299
clutch             0.00098895               0.322313
bearing            0.0003                   0.0977743
Sum                0.306829                 100.
```

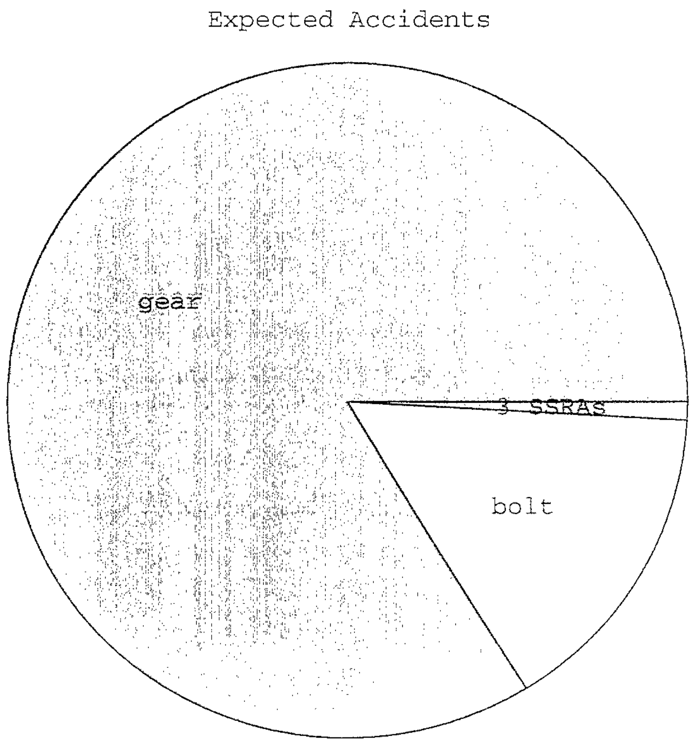
When a single expected-accident column is generated as above, a percentage column is included. The "gear" SSRA is quite dominant and the "bolt" SSRA is also of appreciable magnitude. The same two SSRAs that were dominant in terms of current occurrence rates are also dominant in terms of the number of accidents expected in the next three years.

A pie chart could be generated in lieu of the table above by using the new add-on function `ExpectedAccidentsPie`:

```
?ExpectedAccidentsPie

ExpectedAccidentsPie[SSRAlist, {startDate, endDate}, topSSRAs, opts]
  generates a pie chart of the expected number of accidents
  from startDate to endDate for the SSRAs in SSRAlist. Only
  the largest SSRAs, as specified by topSSRAs, are given their
  own wedges. The remaining SSRAs are placed in a single wedge.
  opts is an optional argument for specifying PieChart options.
```

A pie chart of the expected number of accidents in the next three years, with the top two SSRAs given their own wedges, is generated thus:

```
ExpectedAccidentsPie[ssralist,
  {today, ReplacePart[today, today[[1]] + 3, 1]}, 2, PieStyle -> colors];
```



The three-year expected accidents pie is dramatically different from *today* 's occurrence-rate pie above in that the top two SSRAs have essentially reversed positions.

A table of expected accidents from *today* forward 1, 3 and 5 years for all the SSRAs in *ssralist* is:

```
ExpectedAccidentsTable[ssralist,
  today, {1, 3, 5}, TableSpacing -> {1, 1}] // N
```

```
SSRA           EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
gear           0.0616022            0.257403             0.457403
bolt           0.0446796            0.0467829            0.0467829
solenoidValve  0.00132707           0.00135403           0.00135403
clutch         0.00098895           0.00098895           0.00098895
bearing        0.0001               0.0003               0.0005
SUM            0.108698             0.306829             0.507029
```

It should be noted that `ExpectedAccidentsTable` sorts the SSRAs in decreasing order based on the first column.

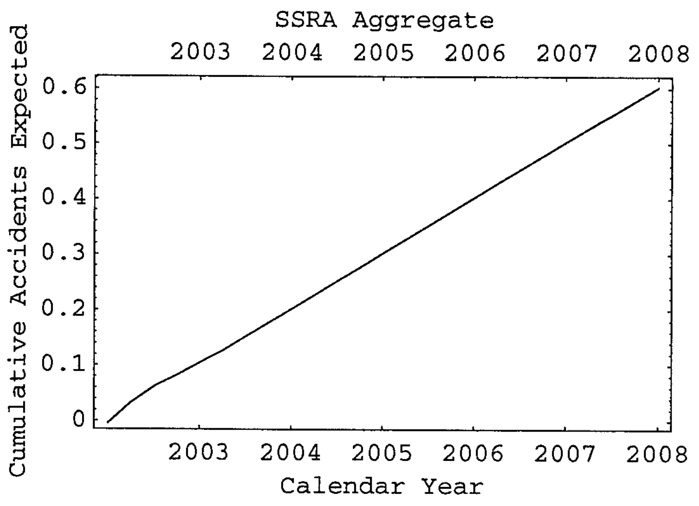
The new add-on function `AggregateExpectedAccidentsPlot` will plot the aggregate of the SSRAs for any time period of interest:

```
? AggregateExpectedAccidentsPlot
```

```
AggregateExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots
  the aggregate of all SSRAs in SSRAlist from a day before
  the earliest begins to a day after the last ends. yearInc
  specifies the calendar-year increments to be used on the
  plot. AggregateExpectedAccidentsPlot[SSRAlist, {startDate,
  endDate}, yearInc, opts] plots the aggregate of all SSRAs from
  startDate to endDate. AggregateExpectedAccidentsPlot[SSRAlist,
  nameList, yearInc, opts] plots the aggregate of only SSRAs
  in nameList from a day before the earliest begins to a day
  after the last ends. AggregateExpectedAccidentsPlot[SSRAlist,
  nameList, {startDate, endDate}, yearInc, opts] plots the
  aggregate of only SSRAs in nameList from startDate to endDate.
  opts is an optional argument for specifying Plot options.
```

The aggregate of the expected number of accidents from *today* forward until the end of 2007 can be plotted thus:

```
aggExpAccPlot = AggregateExpectedAccidentsPlot[ssralist,
   {today, {2007, 12, 31}}, 1, PlotStyle → RGBColor[0, 0, 1]];
```



It should be noted that the plot above is cumulative from *today* forward. This plot was assigned as the value of the symbol *aggExpAccPlot* so that it can be used for comparisons later.

The plot above curves downward slightly since the aggregate occurrence rate decreases which is indicative of risk reduction. Examination of the graphs where the individual SSRAs were plotted reveals that this initial downward curve is due to the "bolt" SSRA. The curve is then quite linear since the aggregate occurrence rate levels off as mentioned above.

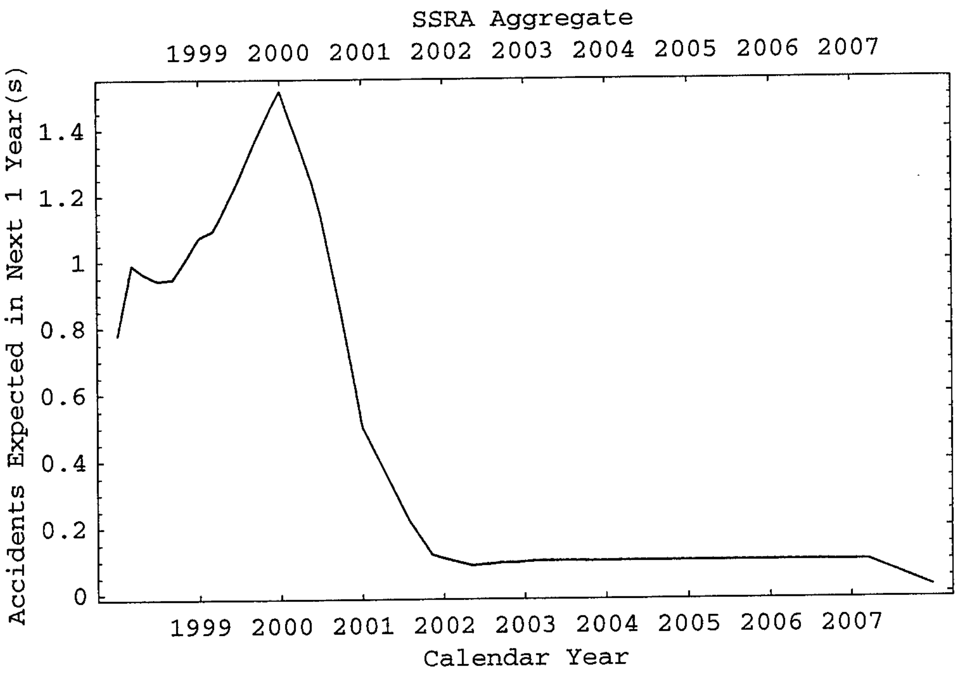
The new add-on function `ExpectedAccidentsTrendPlot` may also be useful:

```
? ExpectedAccidentsTrendPlot
```

```
ExpectedAccidentsTrendPlot[SSRAlist, endDate, year, yearInc,
  opts] plots the n-year aggregate expected-accident trend for
  all SSRAs in SSRAlist from the beginning of the earliest SSRA
  until endDate. yearInc specifies the calendar-year increments
  to be used on the plot. ExpectedAccidentsTrendPlot[SSRAlist,
  {startDate, endDate}, year, yearInc, opts] plots the n-year
  expected accident trend for all SSRAs from startDate to endDate.
  opts is an optional argument for specifying Plot options.
```

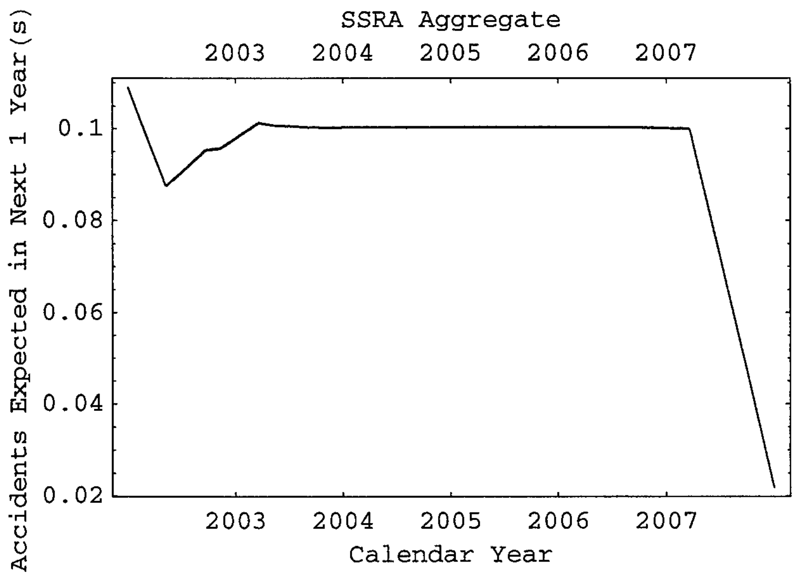
This function will generate a non-cumulative, aggregate expected accidents plot. A plot of the trend in one-year aggregate expected accidents from the beginning of the earliest SSRA until the end of 2007 is:

```
ExpectedAccidentsTrendPlot[ssralist,
  {2007, 12, 31}, 1, 1, PlotStyle → RGBColor[0, 0, 1]];
```



This is a good plot of the historical trend in terms of expected-accident predictions. Incremental risk increased sharply until 2000, then decreased sharply until 2002 and is expected to level off for the next several years. A plot of the trend in one-year aggregate expected accidents from *today* forward until the end of 2007 is:

```
ExpectedAccidentsTrendPlot[ssralist,
  {today, {2007, 12, 31}}, 1, 1, PlotStyle→RGBColor[0, 0, 1]];
```



This plot show some additional details for the next couple of years, principally a slight and brief decrease in the predicted one-year expected-accidents, due to the "bolt" SSRA, followed by a slight and brief increase, due to the "gear" SSRA. The right-most end of the curve should be viewed with caution since the predictions for the notional SSRAs that expire in 2008 will have an impact on one-year expected accident calculations in 2007. This does not necessarily imply that the SSRAs are predicted to be closed in 2008.

It should also be noted that an inspection of the plot above doesn't reveal whether the flight-hours profile is predicted to change and what the impact will be. A plot of the expected accidents in the next 100,000 flight hours will be generated in a subsection below. Comparison of the two trend plots will reveal what the impact of changes to the flight-hour profile is.

Accident probabilities are also important. Numerical values for the aggregate number of accidents can be calculated with the new add-on function `AggregateExpectedAccidents`:

```
? AggregateExpectedAccidents

AggregateExpectedAccidents[SSRAlist, {startDate, endDate}]
  aggregates all SSRAs in SSRAlist from startDate to endDate.
  AggregateExpectedAccidents[SSRAlist, nameList, {startDate, endDate}]
  aggregates only SSRAs in nameList from startDate to endDate.
  AggregateExpectedAccidents[SSRAlist, startDate, years] aggregates
  all SSRAs in SSRAlist from startDate forward the specified number
  of years. AggregateExpectedAccidents[SSRAlist, nameList, startDate,
  years] aggregates only SSRAs in nameList from startDate forward.
```

The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next year is:

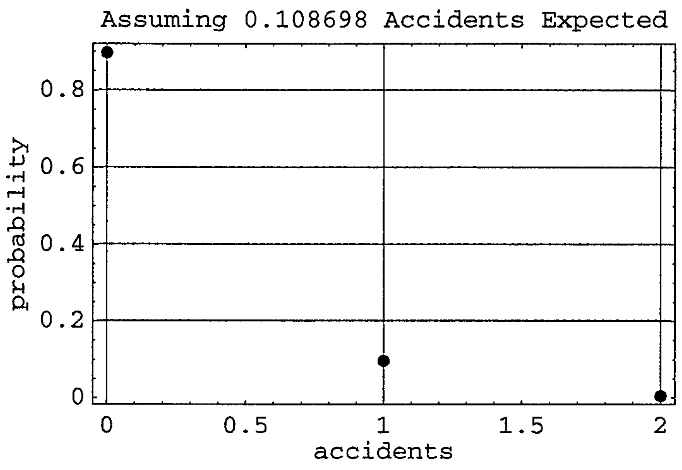
```
AggregateExpectedAccidents[ssralist, today, 1] // N

0.108698
```

Even though this expected number of accidents is quite small, there is a possibility of one or more accidents. The probability of one or more accidents in the next year is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 1]], 0]]

0.102999
```

A plot of the Poisson probability density function will help us visualize these accident probabilities:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%%], x]}, {x, 0, 2}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %%]], None},
  GridLines → {{0, 1, 2}, Automatic}];
```

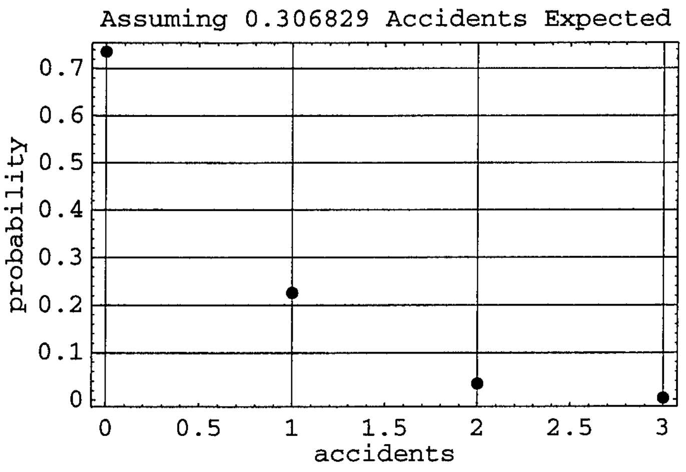


It appears that there's approximately a 10% chance of 1 accident in the next year and very little chance of more than 1. The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next three years is:

```
AggregateExpectedAccidents[ssralist, today, 3] // N

0.306829
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 3}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3}, Automatic}];
```



The probability of one or more accidents in the next three years is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 3]], 0]]

0.264224
```

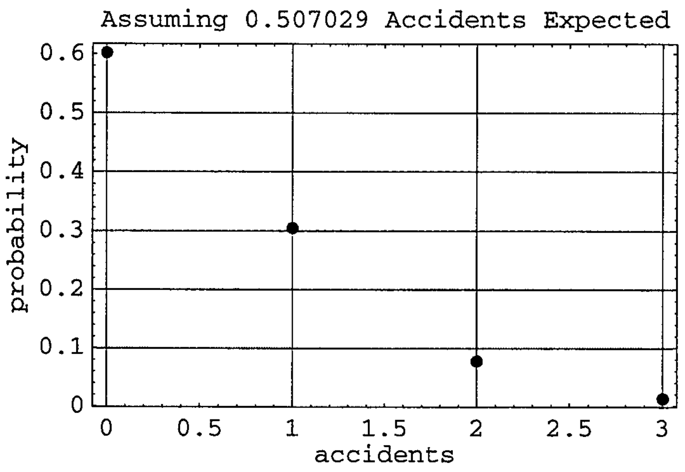
The probability of 1 accident is greater than 20% and the probability of 2 accidents is approximately 5%. There is little chance of more than 2 accidents.

The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next five years is:

```
AggregateExpectedAccidents[ssralist, today, 5] // N

0.507029
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 3}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3}, Automatic}];
```



The probability of one or more severity I helicopter accidents in this notional fleet during the next five years is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 5]], 0]]
```

```
0.397718
```

There's approximately a 30% chance of 1 accident and an 8% chance of 2.

■ Fix the Worst SSRAs?

One may want to consider the impact of fixing the worst SSRAs. It will be helpful to first construct a list of the names of these SSRAs and assign them as the value of the symbol *namelist*:

```
namelist = Map[First[#] &, ssralist]
```

```
{bearing, bolt, clutch, gear, solenoidValve}
```

The worst SSRAs can be found in the expected-accidents table earlier in this analysis of severity-level I SSRAs. In terms of the expected number of accidents in the next three years, there are two worst SSRAs. They can be deleted from *namelist* and this reduced list of names assigned as the value of the symbol *fix3namelist* thus:

```
fix3namelist = Select[namelist, (# ≠ "gear" ∧ # ≠ "bolt") &]

{bearing, clutch, solenoidValve}
```

The expected-accidents table with these SSRAs deleted is:

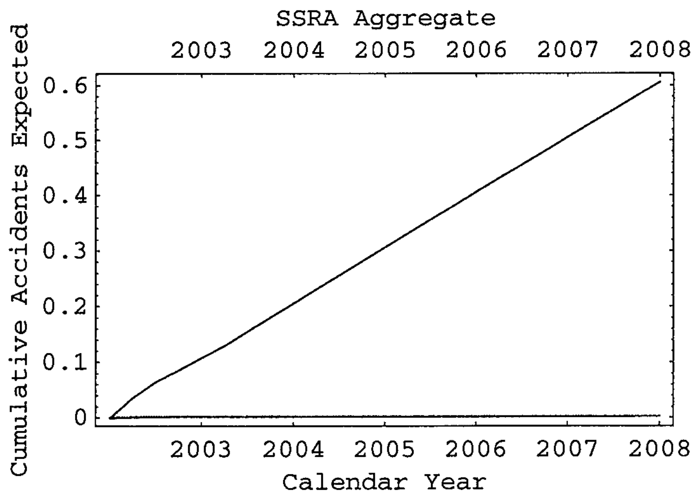
```
ExpectedAccidentsTable[ssralist,
  fix3namelist, today, {1, 3, 5}, TableSpacing → {1, 1}] // N

SSRA          EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
solenoidValve 0.00132707           0.00135403           0.00135403
clutch        0.00098895           0.00098895           0.00098895
bearing       0.0001               0.0003               0.0005
SUM           0.00241602           0.00264298           0.00284298
```

A graph of the aggregate of the expected number of accidents from *today* until the end of 2007, with the two worst SSRAs removed, can be generated but temporarily suppressed thus:

```
fix3Plot = AggregateExpectedAccidentsPlot[
   ssralist, fix3namelist, {today, {2007, 12, 31}}, 1,
   PlotStyle → RGBColor[.5, 0, .5], DisplayFunction → Identity];
```

The plot above is now overlaid on the expected number of accidents plot of all the SSRAs:

```
Show[aggExpAccPlot, fix3Plot];
```



Fixing the two worst SSRAs would dramatically reduce the predicted accident risk. If the worst two SSRAs were fixed *today*, the aggregate expected accidents in the next three years would be decreased to the following percent of the current level:

```
N[AggregateExpectedAccidents[ssralist, fix3namelist, today, 3] /
   AggregateExpectedAccidents[ssralist, today, 3] 100]
```

```
0.861386
```

■ Expected Number of Accidents in Next 100K Flight Hours

It may also be of interest to calculate the number of accidents expected to occur in the next 100,000 hours. Examining the number of accidents expected to occur in the next 100,000 hours provides a mechanism to compare risks of small and large fleets. The new add-on function `FlightHoursDate` will be helpful:

```
?FlightHoursDate
```

```
FlightHoursDate[FHdata, startDate, fhrs}] calculates,
  starting from startDate, the date when fhrs flight hours
  will be accumulated, given the flight-hours data in FHdata.
```

Starting from *today*, the date when 100,000 flight hours will be accumulated is:

```
date100K = FlightHoursDate[HelicopterHours, today, 100000]
```

```
{2003, 4, 22}
```

This date was assigned as the value of the symbol *date100K*. A table of the expected number of accidents between *today* and *date100K* for each of the SSRAs ranked from largest to smallest is:

```
TableForm[Sort[Table[{namelist[[i]],
    N[ExpectedAccidents[ssralist, namelist[[i]], {today, date100K}]]},
   {i, 1, Length[namelist]}], #1[[2]] > #2[[2]] &],
 TableHeadings → {None, {"SSRA", "EXP. ACC. NEXT 100K"}}]
```

```
SSRA                 EXP. ACC. NEXT 100K
gear                 0.0873652
bolt                 0.045884
solenoidValve        0.00135403
clutch               0.00098895
bearing              0.000130056
```

The aggregate expected number of accidents between *today* and *date100K* is:

```
N[prob100K = AggregateExpectedAccidents[ssralist, {today, date100K}]]
```

```
0.135722
```

This quantity could have been directly calculated (i.e., without calculating *date100K* first) using the new add-on function ExpectedAccidentsNext:

```
? ExpectedAccidentsNext
```

```
ExpectedAccidentsNext[SSRAlist, FHdata, startDate, fhrs] calculates,
  starting from startDate, the aggregate number of accidents
  expected to occur in the next fhrs flight hours, given the
  SSRA data in SSRAlist and the flight-hours data in FHdata.
```

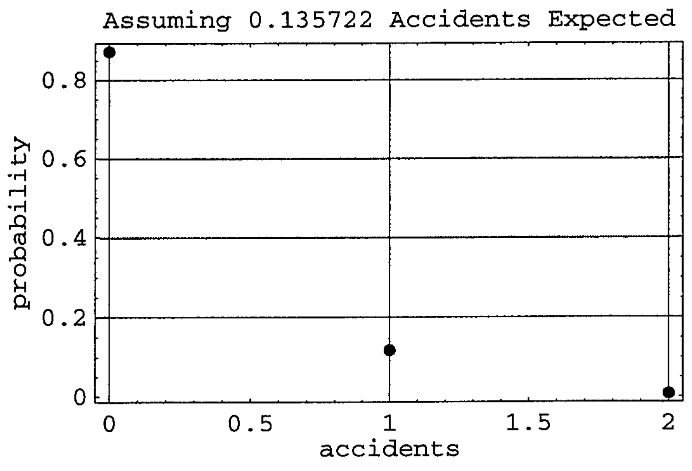
The aggregate expected number of accidents in the next 100,000 flight hours, starting from *today*, is:

```
ExpectedAccidentsNext[ssralist, HelicopterHours, today, 100000] // N
```

```
0.135722
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[prob100K], x]}, {x, 0, 2}],
  FrameLabel → {"accidents", "probability", ToString[
     StringForm["Assuming `1` Accidents Expected", N[prob100K]]], None},
  GridLines → {{0, 1, 2}, Automatic}];
```



The probability of no accidents is:

```
TableForm[Table[{{i}, N[ExpectedAccidentsNext[ssralist,
     HelicopterHours, {i, 1, 1}, 100000]]}, {i, 2001, 2006, 1}],
 TableHeadings → {None, {"CALENDAR YEAR", "EXP. ACC. NEXT 100K"}}]
```

```
CALENDAR YEAR      EXP. ACC. NEXT 100K
     2001               0.560712
     2002               0.135963
     2003                0.12003
     2004               0.123256
     2005               0.132549
     2006               0.135269
```

Analysis of Severity-Level II SSRAs

In this section, the severity-level II SSRAs are analyzed. Since the most recent SSRA falls into this category, it is addressed as well. Finally, the impact of immediately fixing the worst SSRAs is considered.

Aggregation and Analysis

First, the severity-level II SSRAs are extracted from the complete list of SSRAs and assigned as the value of the symbol *ssralist*:

```
ssralist = SSRASelect[HelicopterSSRAs, "II"];
```

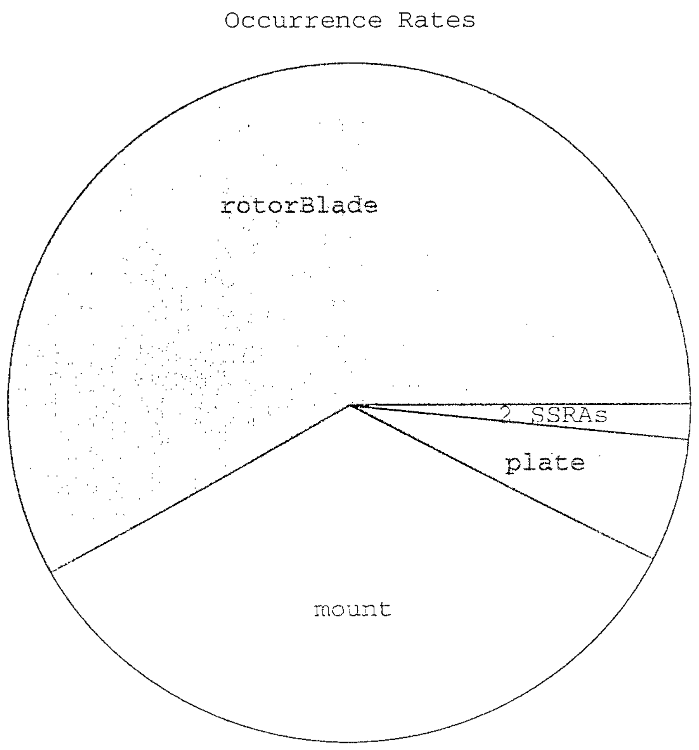
A table of *today*'s occurrence rates for the SSRAs in *ssralist*, in decreasing order, is generated as follows:

```
OccurrenceRateTable[ssralist, today] // N
```

```
SSRA              OCCURRENCE RATE (ACCIDENTS/DAY)      PERCENT
rotorBlade        0.000274725                          58.091
mount             0.000163043                          34.4758
plate             0.0000274725                         5.8091
solenoid          4.41989×10^-6                        0.934592
shutoffValve      3.26087×10^-6                        0.689515
SUM               0.000472922                          100.
```

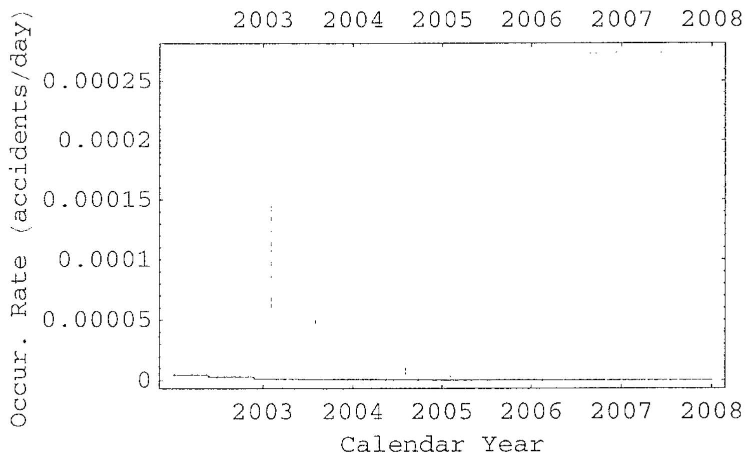
The first two SSRAs, "rotorBlade" and "mount", are quite dominant and the third is also of appreciable magnitude. A pie chart of *today*'s occurrence rates, with the top three SSRAs given their own wedges, is generated thus:

```
OccurrenceRatePie[ssralist, today, 3, PieStyle → colors];
```



The table and pie chart above represent only a snapshot in time. A plot of the occurrence rates of all the SSRAs in *ssralist* from *today* until the end of 2007 is generated thus:

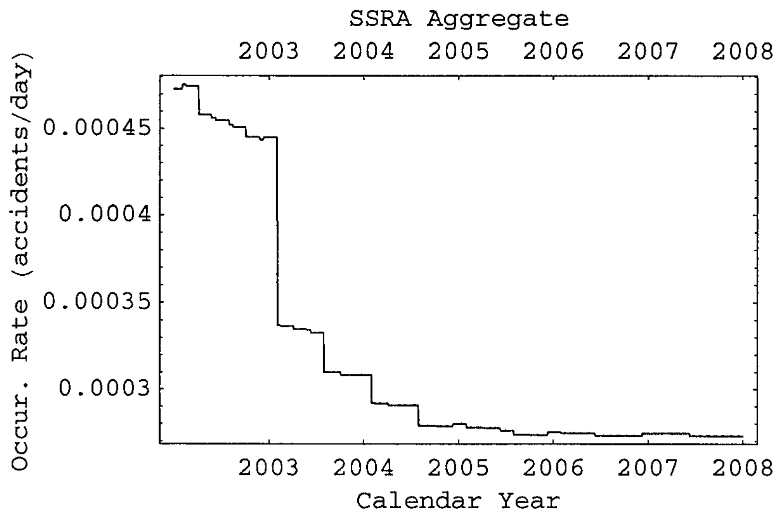
```
OccurrenceRatePlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → colors];
```



The occurrence rate for the "rotorBlade" SSRA is initially uppermost. It's level until 2008. The occurrence rate for the "mount" SSRA, the second most dominant, is expected to decrease starting in 2003. This decrease is due presumably to the attrition of weak components, improved risk-mitigation and/or the declining fleet size. The "plate" SSRA, the third most dominant, is expected to quickly drop into the noise. Additional risk mitigation efforts should focus primarily on the "rotorBlade" SSRA.

A plot of the aggregate of the SSRA occurrence rates above from *today* until the end of 2007 is:

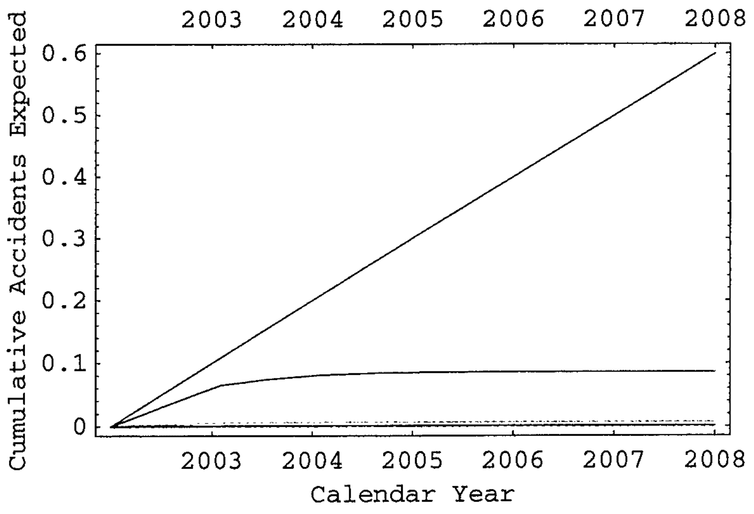
```
AggregateOccurrenceRatePlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → RGBColor[0, 0, 1]];
```



The aggregated occurrence rate follows the pattern of the dominant SSRAs, as expected. The initial decrease is due to both the "mount" and "plate" SSRAs. The increase thereafter is due entirely to the "mount" SSRA.

The expected number of accidents for all the SSRAs in *ssralist* from *today* until the end of 2007 is:

```
ExpectedAccidentsPlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → colors];
```



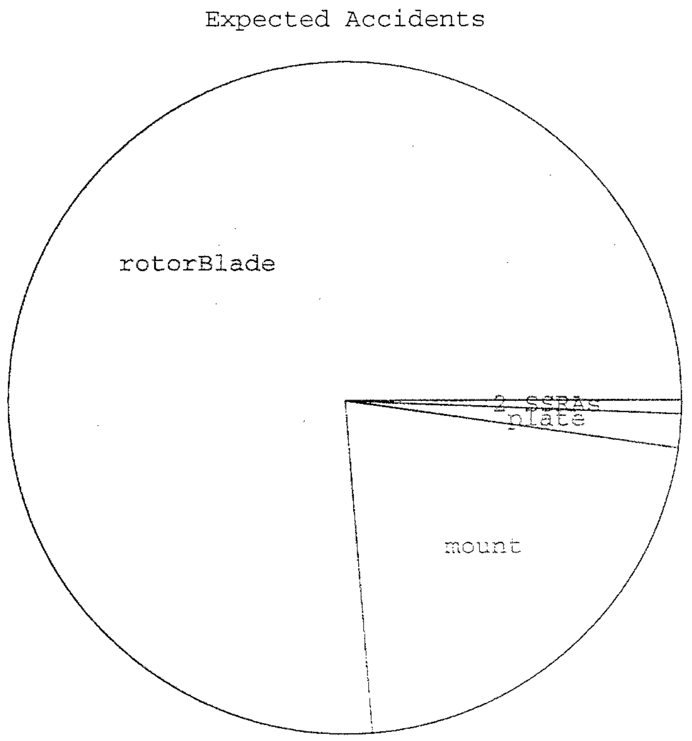
The cumulative expected-accident curve for the "rotorBlade" SSRA is quite linear indicating that the incremental risk is getting neither better or worse. The plot of the "mount" SSRA is curving downward since the occurrence rate (and thus the incremental risk) is decreasing over time as noted above.

A table of expected-accident values from *today* forward 1, 3 and 5 years for all the SSRAs in *ssralist* can be generated as follows:

```
ExpectedAccidentsTable[ssralist,
  today, {1, 3, 5}, TableSpacing → {1, 1}] // N

SSRA          EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
rotorBlade    0.1                  0.3                  0.5
mount         0.06                 0.0844864            0.0852283
plate         0.00497253           0.00646209           0.00652747
solenoid      0.00117072           0.00155978           0.00156762
shutoffValve  0.000693478          0.00092788           0.000946984
SUM           0.166837             0.393436             0.59427
```

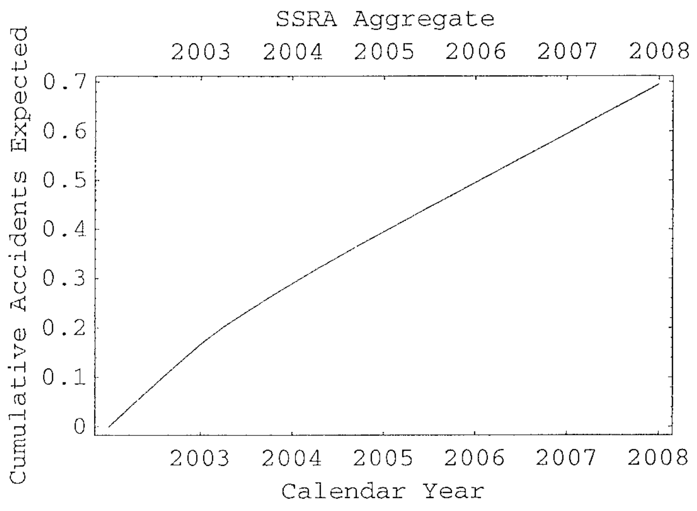
A pie chart of the expected number of accidents during the next three years, the center column above, with the top three SSRAs given their own wedges, is generated thus:

```
ExpectedAccidentsPie[ssralist,
  {today, ReplacePart[today, today[[1]] + 3, 1]}, 3, PieStyle → colors];
```



It can be seen that the "plate" SSRA is dropping into the noise. A cumulative plot of the aggregate of the expected number of accidents from *today* forward until the end of 2007 can be plotted thus:

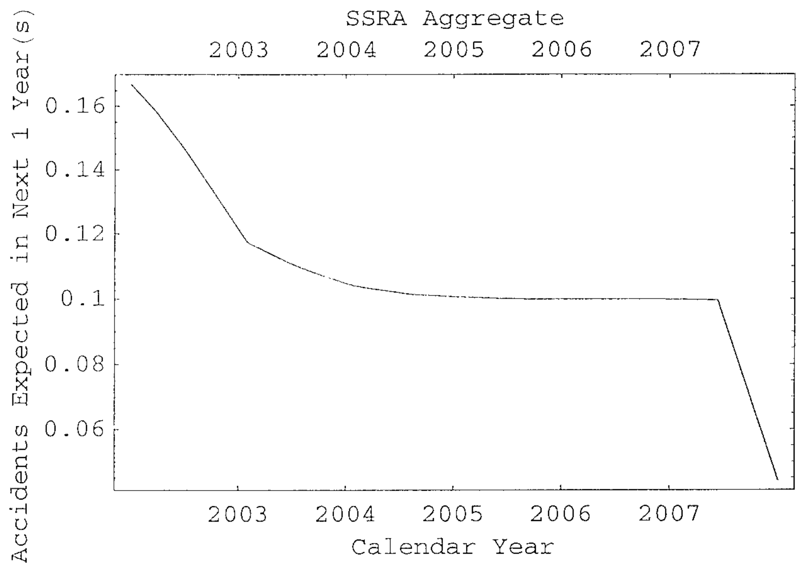
```
aggExpAccPlot = AggregateExpectedAccidentsPlot[ssralist,
   {today, {2007, 12, 31}}, 1, PlotStyle → RGBColor[0, 0, 1]];
```



The cumulative aggregate plot curves due to the "mount" SSRA.

A non-cumulative aggregate expected accidents plot may be helpful. A plot of the trend in one-year aggregate expected accidents from *today* forward until the end of 2007 is:

```
ExpectedAccidentsTrendPlot[ssralist,
  {today, {2007, 12, 31}}, 1, 1, PlotStyle → RGBColor[0, 0, 1]];
```



This plot show a strong decrease in the one-year expected accidents trend during the next year due to both the "mount" and "plate" SSRAs. A more gradual decrease is expected to occur between 2003 and 2005 due to the "mount" SSRA. Improvement in this single SSRA generates excellent progress for the SSRA aggregate. The decline that begins in 2007 should be viewed with skepticism as it may be due to expiration of SSRA predictions, not the SSRAs themselves.

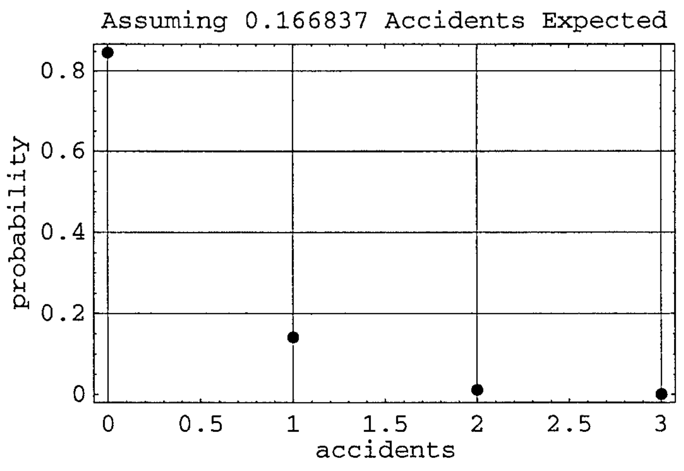
It should also be noted that an inspection of the plot above doesn't reveal whether the flight-hours profile is predicted to change and what the impact will be. A plot of the expected accidents in the next 100,000 flight hours will be generated in a subsection below. Comparison of the two trend plots will reveal what the impact of changes to the flight-hour profile is.

Accident probabilities are also important. The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next year is:

```
AggregateExpectedAccidents[ssralist, today, 1] // N

0.166837
```

A plot of the Poisson probability density function will provide accident probabilities:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 3}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3}, Automatic}];
```



The probability of having no accidents in the next year is approximately 0.85. The probability of one or more accidents in the next year is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 1]], 0]]
```

```
0.153662
```

It's clear from the graph above that the probability of two or more accidents is quite small.

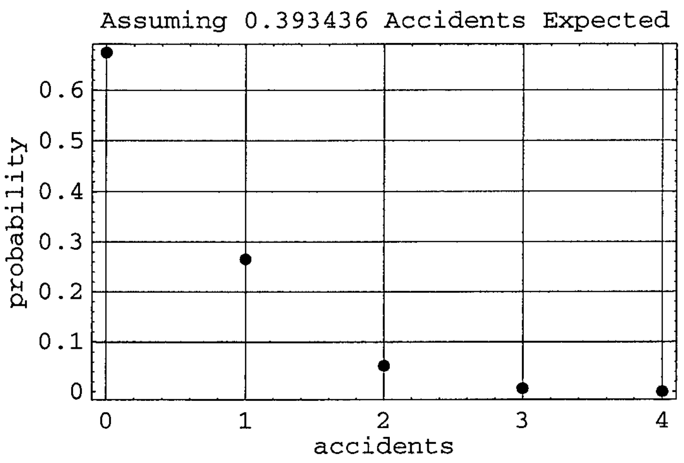
The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next three years is:

```
AggregateExpectedAccidents[ssralist, today, 3] // N
```

```
0.393436
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 4}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3, 4}, Automatic}];
```



The probability of no accidents in the next three years is approximately 0.67. The probability of one or more accidents is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 3]], 0]]
```

```
0.325266
```

There's approximately a 27% chance of exactly 1 accident and a 6 % chance of exactly 2 accidents. There's little chance of 3 or more accidents.

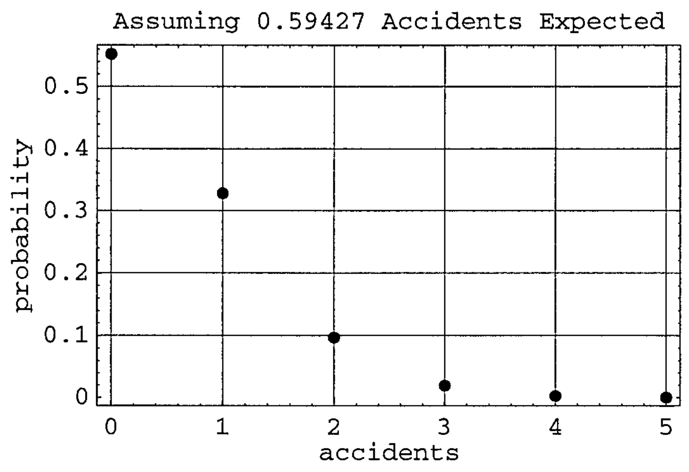
The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next five years is:

```
AggregateExpectedAccidents[ssralist, today, 5] // N
```

```
0.59427
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 5}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3, 4, 5}, Automatic}];
```



The probability of no accidents is approximately 0.55. The probability of one or more severity II accidents in the next five years is:

```
N[1 - PDF[PoissonDistribution[
    AggregateExpectedAccidents[ssralist, today, 5]], 0]]
```

```
0.448035
```

The chance of exactly 1, 2 or 3 accidents is approximately 33, 10, and 2%, respectively.

■ Impact of Most Recent SSRA

The last SSRA to begin (obtained earlier in this chapter) was "shutoffValve". It may be useful to generate key risk-aggregation results as in the subsection above but with this SSRA deleted. This will permit one to perform a before-and-after assessment for this SSRA.

It will be helpful to first construct a list of the names of these SSRAs and assign them as the value of the symbol *namelist*:

```
namelist = Map[First[#] &, ssralist]
```

```
{mount, plate, rotorBlade, shutoffValve, solenoid}
```

One can generate the expected-accidents table without the last SSRA by first deleting it from *namelist* and assigning this reduced list of names as the value of the symbol *priornamelist* thus:

```
priornamelist = Select[namelist, # ≠ "shutoffValve" &]
```

```
{mount, plate, rotorBlade, solenoid}
```

The expected-accidents table without the last SSRA is then:

```
ExpectedAccidentsTable[ssralist,
  priornamelist, today, {1, 3, 5}, TableSpacing → {1, 1}] // N
```

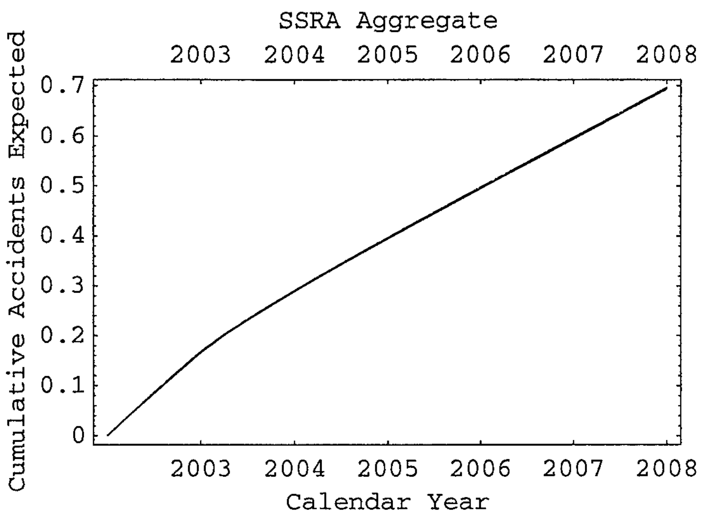
```
SSRA        EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
rotorBlade  0.1                  0.3                  0.5
mount       0.06                 0.0844864            0.0852283
plate       0.00497253           0.00646209           0.00652747
solenoid    0.00117072           0.00155978           0.00156762
SUM         0.166143             0.392508             0.593323
```

A graph of the aggregate of the expected number of accidents from *today* forward until the end of 2007, with the most recent SSRA removed, can be generated and temporarily suppressed thus:

```
beforePlot = AggregateExpectedAccidentsPlot[
   ssralist, priornamelist, {today, {2007, 12, 31}}, 1,
   PlotStyle → RGBColor[1, 0, 0], DisplayFunction → Identity];
```

The plot above is now overlaid on the expected number of accidents plot of all the SSRAs:

```
Show[aggExpAccPlot, beforePlot];
```



The new curve overlays the first since the "shutoffValve" SSRA had a very minor impact. The aggregate number of expected accidents over the next three years increased after the last SSRA by a factor of:

```
N[AggregateExpectedAccidents[ssralist, today, 3] /
  AggregateExpectedAccidents[ssralist, priornamelist, today, 3]]
```

```
1.00236
```

- **Fix the Worst SSRAs?**

One may also want to consider the impact of fixing the worst SSRAs. The worst SSRAs can be found in the expected-accidents table earlier in this analysis of severity-level II SSRAs. In terms of the expected number of failures in the next three years, there are two worst SSRAs. They can be deleted from *namelist* and this reduced list of names assigned as the value of the symbol *fix3namelist* thus:

```
fix3namelist = Select[namelist, (# ≠ "rotorBlade" ∧ # ≠ "mount") &]
```

```
{plate, shutoffValve, solenoid}
```

The expected-accidents table with these SSRAs deleted is:

```
ExpectedAccidentsTable[ssralist,
  fix3namelist, today, {1, 3, 5}, TableSpacing → {1, 1}] // N
```

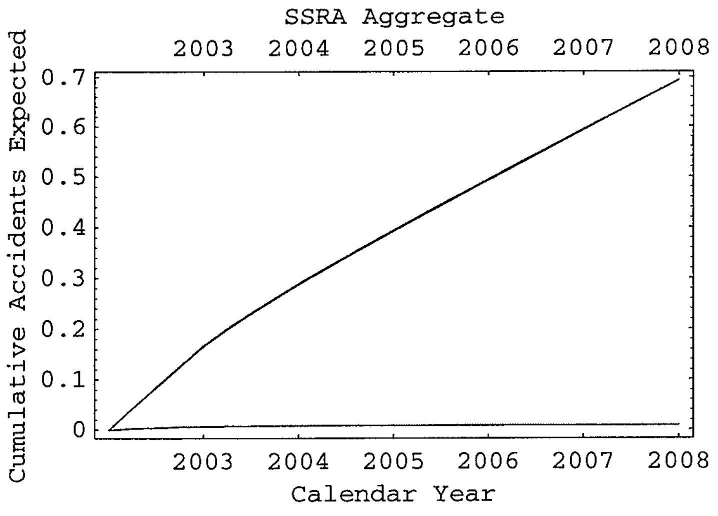
```
SSRA          EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
plate         0.00497253           0.00646209           0.00652747
solenoid      0.00117072           0.00155978           0.00156762
shutoffValve  0.000693478          0.00092788           0.000946984
SUM           0.00683672           0.00894975           0.00904208
```

A graph of the aggregate expected number of accidents from *today* forward until the end of 2007, with the three worst SSRAs removed, can be generated but temporarily suppressed thus:

```
fix3Plot = AggregateExpectedAccidentsPlot[
   ssralist, fix3namelist, {today, {2007, 12, 31}}, 1,
   PlotStyle → RGBColor[.5, 0, .5], DisplayFunction → Identity];
```

The plot above is now overlaid on the expected number of accidents plot of all the SSRAs and with the last SSRA removed:

```
Show[aggExpAccPlot, beforePlot, fix3Plot];
```



It can be seen that fixing the two worst SSRAs will result in a dramatic improvement. If the worst two SSRAs were fixed *today*, the aggregate expected accidents in the next three years would be decreased to the following percent of the current level:

```
N[AggregateExpectedAccidents[ssralist, fix3namelist, today, 3] /
   AggregateExpectedAccidents[ssralist, today, 3] 100]
```

```
2.27476
```

■ Expected Number of Accidents in Next 100K Flight Hours

It may also be of interest to calculate the number of accidents expected to occur in the next 100,000 hours. In the section on severity-level I SSRAs, the date when the fleet will accumulate 100,000 hours was assigned as the value of the symbol *date100K*:

```
date100K
```

```
{2003, 4, 22}
```

A table of the expected number of accidents between *today* and *date100K* for each of the SSRAs is:

```
TableForm[Sort[Table[{namelist[[i]],
    N[ExpectedAccidents[ssralist, namelist[[i]], {today, date100K}]]},
   {i, 1, Length[namelist]}], #1[[2]] > #2[[2]] &],
 TableHeadings → {None, {"SSRA", "EXP. ACC. NEXT 100K"}}]
```

```
SSRA             EXP. ACC. NEXT 100K
rotorBlade       0.129945
mount            0.0691482
plate            0.00548321
solenoid         0.00129116
shutoffValve     0.000773035
```

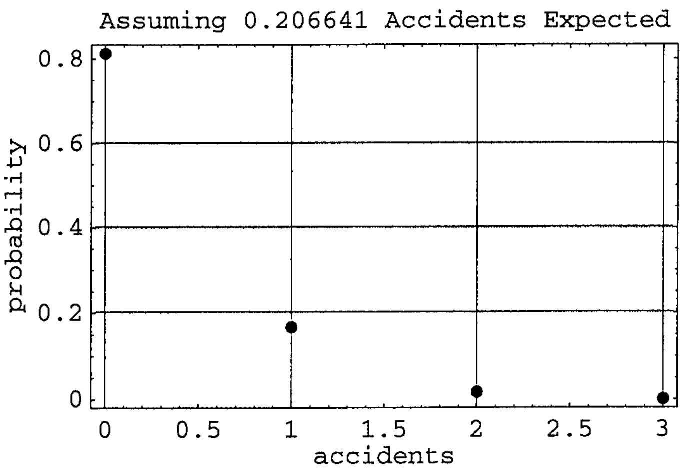
The aggregate expected number of accidents between *today* and *date100K* is:

```
N[prob100K = AggregateExpectedAccidents[ssralist, {today, date100K}]]
```

```
0.206641
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[prob100K], x]}, {x, 0, 3}],
  FrameLabel → {"accidents", "probability", ToString[
     StringForm["Assuming `1` Accidents Expected", N[prob100K]]], None},
  GridLines → {{0, 1, 2, 3}, Automatic}];
```



The probability of no accidents is:

```
PDF[PoissonDistribution[prob100K], 0] // N
```

```
0.813312
```

The probability of one or more accidents is:

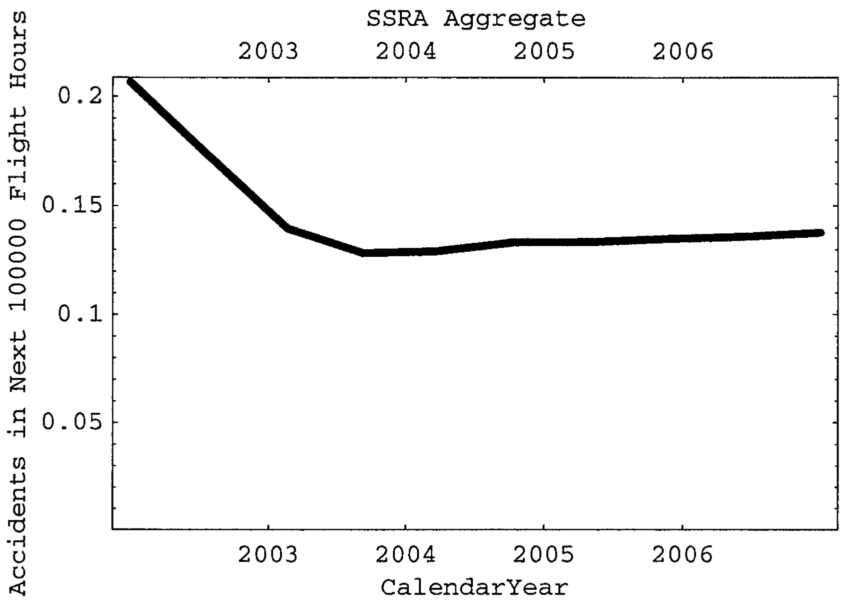
```
N[1 - PDF[PoissonDistribution[prob100K], 0]]
```

```
0.186688
```

There is little chance of two or more accidents occurring.

The number of accidents expected to occur in the next 100,000 flight hours from *today* forward is plotted thus:

```
ExpectedAccidentsNextPlot[ssralist,
  HelicopterHours, {today, {2006, 12, 31}}, 100000, 1];
```



Comparison of this plot to the accidents-expected-in-the-next-year plot earlier reveals that this plot shows that the patterns are quite similar. There is a modest increase in 2004 whereas the earlier plot was level. This indicates that the accidents-expected-in-the-next-year plot would increase during this time period if not for the effect of a declining flight-hours profile. Comparison of these two graphs can be quite useful.

A table of the expected number of accidents in the next 100,000 flights hours, starting from the beginning of calendar years 2001 through 2006, can be generated thus:

```
TableForm[Table[{{i}, N[ExpectedAccidentsNext[ssralist,
     HelicopterHours, {i, 1, 1}, 100000]]}, {i, 2001, 2006, 1}],
 TableHeadings → {None, {"CALENDAR YEAR", "EXP. ACC. NEXT 100K"}}]
```

```
CALENDAR YEAR     EXP. ACC. NEXT 100K
    2001               0.17757
    2002               0.206917
    2003               0.145824
    2004               0.128398
    2005               0.13326
    2006               0.135187
```

Analysis of Severity-Level I and II SSRAs

In this section, the severity-level I and II SSRAs will be combined without weighting and then analyzed. This section does not include the "what ifs" regarding the impact of the most recent SSRA and fixing the worst SSRAs, nor does it include the subsection on the expected number of accidents in the next 100,000 flight hours. This was done in the interests of brevity. In order to perform these analyses, one could simply follow the examples in the previous section.

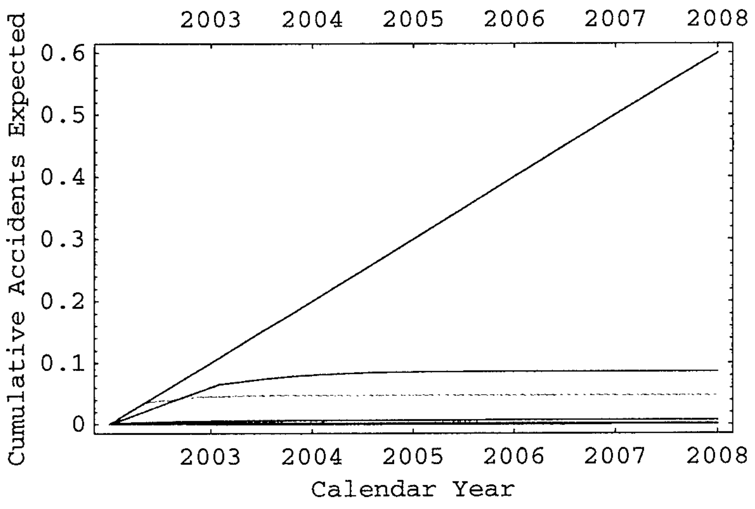
▪ Aggregation and Analysis

First, the severity-level I and II SSRAs for the notional helicopter fleet are extracted from the complete list of SSRAs and assigned as the value of the symbol *ssralist*:

```
ssralist = Join[SSRASelect[HelicopterSSRAs, "I"],
    SSRASelect[HelicopterSSRAs, "II"]];
```

The expected number of accidents for all the SSRAs in *ssralist* from *today* until the end of 2007 is:

```
ExpectedAccidentsPlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle → colors];
```



The cumulative expected-accident curves for the dominant SSRAs curve downward due presumably to the attrition of weak components, improved risk mitigation and/or the declining fleet size.

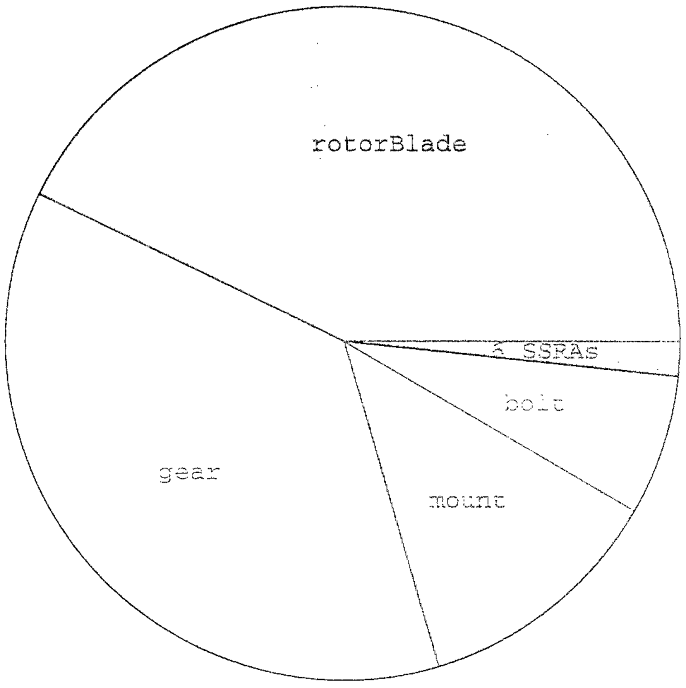
A table of expected accidents from *today* forward 1, 3 and 5 years for all the SSRAs in *ssralist* is:

```
ExpectedAccidentsTable[ssralist,
  today, {1, 3, 5}, TableSpacing → {1, 1}] // N

SSRA           EXP. ACC. 1 YEAR(s)  EXP. ACC. 3 YEAR(s)  EXP. ACC. 5 YEAR(s)
rotorBlade     0.1                  0.3                  0.5
gear           0.0616022            0.257403             0.457403
mount          0.06                 0.0844864            0.0852283
bolt           0.0446796            0.0467829            0.0467829
plate          0.00497253           0.00646209           0.00652747
solenoidValve  0.00132707           0.00135403           0.00135403
solenoid       0.00117072           0.00155978           0.00156762
clutch         0.00098895           0.00098895           0.00098895
shutoffValve   0.000693478          0.00092788           0.000946984
bearing        0.0001               0.0003               0.0005
SUM            0.275535             0.700265             1.1013
```

Upon review of the table above, the "rotorBlade" and "gear" SSRAs are quite dominant. The "mount" and "bolt" SSRAs are also of appreciable magnitude. The remaining six SSRAs are in the noise. A pie chart might help here. A pie chart of the expected number of accidents during the next three years, the center column above, with the top four SSRAs given their own wedges, is generated thus:

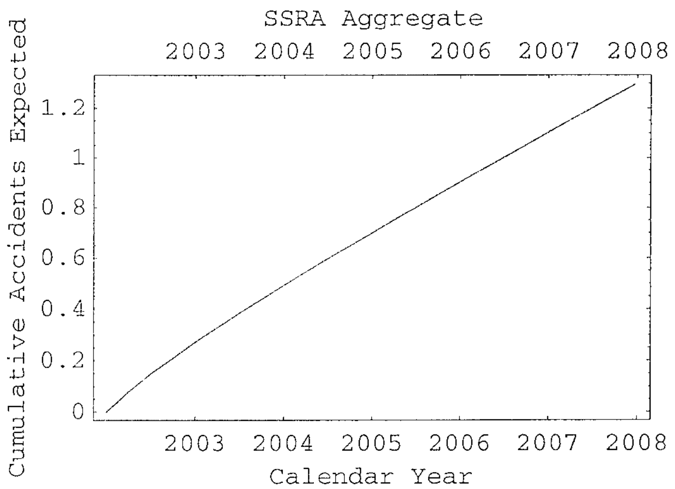
```
ExpectedAccidentsPie[ssralist,
  {today, ReplacePart[today, today[[1]] + 3, 1]}, 4, PieStyle → colors];
```

Expected Accidents



A cumulative plot of the aggregate of the expected number of accidents from *today* forward until the end of 2007 can be plotted thus:

```
AggregateExpectedAccidentsPlot[ssralist,
  {today, {2007, 12, 31}}, 1, PlotStyle→RGBColor[0, 0, 1]];
```



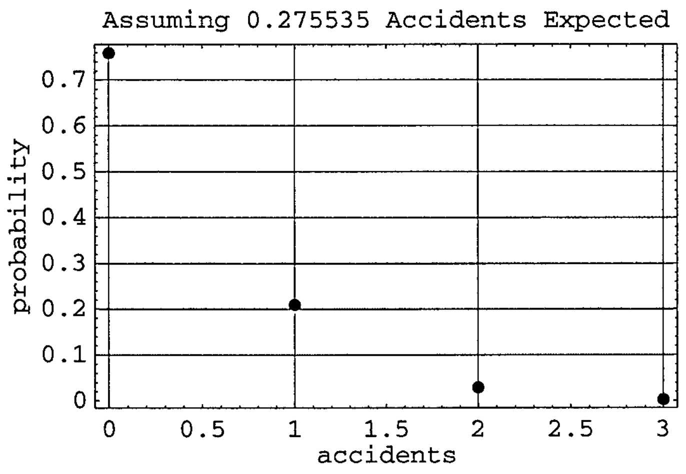
The cumulative aggregate plot curves downward thereby following the dominant SSRAs for the reasons specified above. This is indicative of declining incremental risk.

Accident probabilities are also important. The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next year is:

```
AggregateExpectedAccidents[ssralist, today, 1] // N

0.275535
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 3}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3}, Automatic}];
```



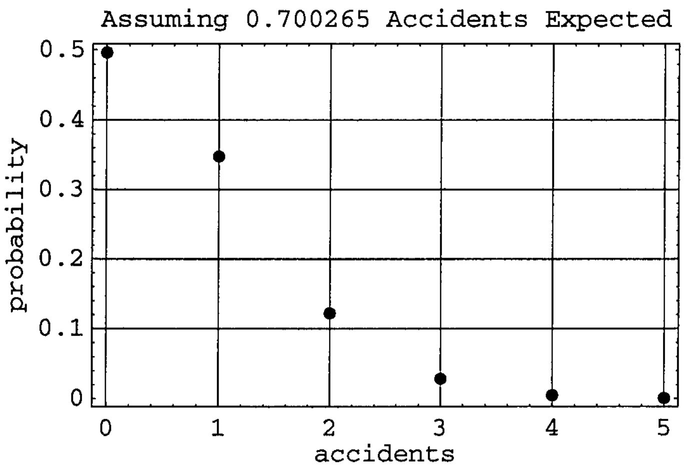
The probability of having no accidents in the next year is approximately 0.76. The probability of exactly 1 and 2 accidents is approximately 21 and 3%, respectively. There is very little chance of 3 or more accidents.

The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next three years is:

```
AggregateExpectedAccidents[ssralist, today, 3] // N

0.700265
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 5}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3, 4, 5}, Automatic}];
```



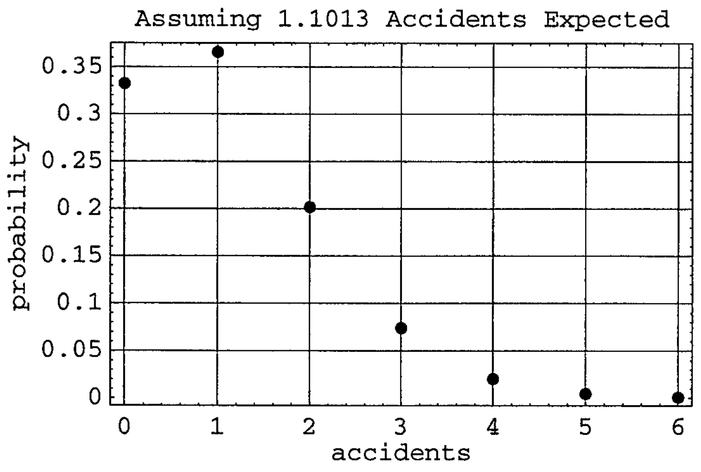
The probability of having no accidents in the next three years is approximately 0.5. There is little chance of 4 or more accidents.

The aggregate expected number of accidents for all the SSRAs in *ssralist* in the next five years is:

```
AggregateExpectedAccidents[ssralist, today, 5] // N

1.1013
```

The corresponding plot of accident probabilities is:

```
ListPlot[Table[{x, PDF[PoissonDistribution[%], x]}, {x, 0, 6}],
  FrameLabel → {"accidents", "probability",
    ToString[StringForm["Assuming `1` Accidents Expected", %]], None},
  GridLines → {{0, 1, 2, 3, 4, 5, 6}, Automatic}];
```



The possibility that no accidents will occur is not the most likely outcome. The occurrence of exactly one accident is the most likely outcome. The probability of one or more severity I or II accidents for the notional helicopter fleet during the next five years is:

```
1 - PDF[PoissonDistribution[%%], 0]

0.667561
```

Summary

This chapter contains the aggregation and analysis of the SSRAs for the notional helicopter fleet using the EAA model. Severity-level I and II SSRAs were analyzed both separately and in combination (with equal weighting). Many graphs and tables were generated for each of the severity levels and combinations thereof. This chapter is an example of how one might go about aggregating and analyzing SSRAs for actual helicopter fleets.

Tables and pie charts were generated for the accident occurrence rates at present. Individual and aggregate SSRA occurrence rates were plotted to show their incremental risk over time. Tables and pie charts were generated for the expected number of accidents at particular points in time. The expected number of accidents was aggregated over various time periods and presented in a variety of ways. Plots of the next year and next 100,000 flight hours expected-accidents trends were also generated and compared. The impacts of the most recent SSRA and fixing the worst SSRAs were considered. Accident probabilities were also calculated and plotted. Insights for the notional SSRAs are included as well. The various graphics generated in this chapter help one put an SSRA into context, not just now, but over time.

How the "riskiness" of an SSRA changes with time was illustrated repeatedly. It's important to see an SSRA within the context of all the SSRAs for a given helicopter fleet over time. An SSRA that may appear to be a top priority at present may be eclipsed when one looks a couple of years into the future. This was illustrated with the notional example. SSRAs should be re-categorized periodically based on their short- or mid-term risks.

Chapter 5

Summary

AMSAA developed two models that aggregate the risks due to the collection of open SSRAs applicable to a helicopter fleet at a point in time. These models can be used to calculate accident probabilities due to SSRA causes as SSRAs are identified and resolved through time. This will allow decision makers to consider the cumulative risk to the fleet due to all open SSRAs when deciding which risk-mitigation option to select for a new SSRA.

Chapter 2 provided background on the need for a new model and describes the development of two models for aggregating SSRAs. The first, more complex model is termed the Component-Socket Hazard model. This model can provide helicopter-level accident predictions as a function of time at the cost of significant computational complexity. Unfortunately, it requires component aging models and component-age data, information not currently available for all SSRAs. It is possible to apply this model in the future when the requisite data becomes available.

The EAA model aggregates the expected number of yearly accidents for each SSRA over a multi-year time horizon. This model addresses only fleet level risk, and not the individual risk for each helicopter. However, the data requirements for this model match the current state of data collection. AMSAA has developed two implementations of this model, one in *Mathematica* and the other in *Excel.* The *Mathematica* functions for the EAA model may be found in Appendix A. The use of these new functions, in combination with the built-in *Mathematica* functions, was illustrated on notional data in Chapters 3 and 4. These chapters may be used as templates for application of the EAA model to actual helicopter fleets. AMSAA Technical Report 695 documents the application of the EAA model to the Apache fleet using these templates.

In Chapter 4, the expected number of accidents was aggregated over various time periods and presented in a variety of ways. Plots of the next year and next 100,000 flight hours expected-accidents trends were also generated and compared. The impacts of the most recent SSRA and fixing the worst SSRAs were considered as well. Accident probabilities were calculated and plotted using a mix of built-in and add-on functions already available in *Mathematica*. The various graphics generated with the EAA model help one put an SSRA into context, not just now, but over time. How the "riskiness" of an SSRA changes with time was illustrated with the notional example in Chapters 3 and 4. It's important to see an SSRA within the context of all the SSRAs for a given helicopter fleet over time. An SSRA that may appear to be a top priority at present may be eclipsed when one looks a couple of years into the future.

THIS PAGE INTENTIONALLY LEFT BLANK.

References

Barlow, R.E., *Engineering Reliability*, ASA and SIAM, Philadelphia, PA, 1998.

Barlow, R.E., Proschan, F., *Statistical Theory of Reliability and Life Testing: Probability Models*, To Begin With, Silver Spring, MD, 1981.

Cushing, M.J., "Application of the Expected Accident Aggregation Model to the Apache Helicopter", US Army Materiel Systems Analysis Activity Technical Report 695, Aberdeen Proving Ground, MD, December 2001.

Kumamoto, H., Henley, E.J., *Probabilistic Risk Assessment and Management for Engineers and Scientists*, 2nd ed., IEEE Press, New York, 1996.

Meeker, W.Q., Escobar, L.A., *Statistical Methods for Reliability Data*, John Wiley & Sons, New York, 1998.

McCullough, B.D., "The accuracy of *Mathematica* 4 as a statistical package", *Computational Statistics 15*, pp. 279-299, 2000.

Wolfram, S., *The Mathematica Book*, 4th ed., Wolfram Media/Cambridge University Press, 1999.

THIS PAGE INTENTIONALLY LEFT BLANK.

Appendix A

Mathematica Package for Expected Accident Aggregation Model

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix A

Mathematica Package for Expected Accident Aggregation Model

This notebook contains functions for the Expected Accident Aggregation model.

Reference

Title

Package for Expected Accident Aggregation Model

Author

Michael J. Cushing, Ph.D.

Summary

This notebook contains functions for the Expected Accident Aggregation model.

Copyright

Not copyrighted.

Notebook Version

See package usage message.

Mathematica Version

4.0

History

This package is new.

Keywords

Cumulative aviation risk, Expected Accident Aggregation Model.

Source

The format for this package is based on the template proposed in Maeder, R. *Programming in Mathematica*, 3rd ed. Addison-Wesley, 1996.

Warnings

Note: all cells marked as "InitializationCell" (i.e., all the executable cells) will be written to the Auto-Save package whenever this notebook is saved. This package can then be read in programs that use it with **`Needs["AviationSafety`ExpectedAccidentAggregationModel`"]`**. Cells not intended to belong to the package do not have this property.

Limitation

None.

Discussion

A potential enhancement would be to modify the plotting functions (especially `ExpectedAccidents-Plot` and `AggregateExpectedAccidentsPlot` so that *endDate* could be specified in terms of years after *startDate*. Another enhancement would be to incorporate additional checking and trapping of input errors.

Requirements

AviationSafety`ExpectedAccidentAggregationModel`

Interface

This part declares the publicly visible functions, options, and values.

- **Set up the package context, including public imports**

```
BeginPackage["AviationSafety`ExpectedAccidentAggregationModel`",
  "Miscellaneous`Calendar`", "Graphics`Graphics`"]
```

- **Usage messages for the exported functions and the context itself**

```
ExpectedAccidentAggregationModel::usage =
 "ExpectedAccidentAggregationModel.m (version 1.0.1) is a package that
   contains functions for the Expected Accident Aggregation model."
```

```
SSRATest::"usage" =
 "SSRATest[ssra] tests an SSRA to ensure it's structured correctly."


SSRASelect::"usage" = "SSRASelect[SSRAlistAll, sevlev]
   selects all SSRAs in SSRAlistAll with severity level sevlev."


FirstSSRA::"usage" =
 "FirstSSRA[SSRAlist] selects from SSRAlist the SSRA that began first."


LastSSRA::"usage" =
 "LastSSRA[SSRAlist] selects from SSRAlist the SSRA that began last."


OccurrenceRate::"usage" =
 "OccurrenceRate[SSRAlist, name, date] calculates
   the occurrence rate (accidents per day) on the
   specified date for the named SSRA from SSRAlist."


OccurrenceRateTable::"usage" =
 "OccurrenceRateTable[SSRAlist, date, opts] generates a table
   of occurrence rates in accidents per day on the specified
   date for all SSRAs in SSRAlist. OccurrenceRateTable[
   SSRAlist, nameList, date, opts] generates a table of
   occurrence rates for only SSRAs in nameList. opts is
   an optional argument for specifying TableForm options."


OccurrenceRatePie::"usage" =
 "OccurrenceRatePie[SSRAlist, date, topSSRAs, opts] generates
   a pie chart of occurrence rates in accidents per day on the
   specified date for the SSRAs in SSRAlist. Only the largest
   SSRAs, as specified by topSSRAs, are given their own wedges.
   The remaining SSRAs are placed in a single wedge.  opts
   is an optional argument for specifying PieChart options."
```

```
OccurrenceRatePlot::"usage" =
 "OccurrenceRatePlot[SSRAlist, yearInc, opts] plots the occurrence
   rates (accidents per day) of all SSRAs in SSRAlist from a
   day before the earliest begins to a day after the last ends.
   yearInc specifies the calendar-year increments to be used on
   the plot. OccurrenceRatePlot[SSRAlist, {startDate, endDate},
   yearInc, opts] plots the occurrence rates of all SSRAs from
   startDate to endDate. OccurrenceRatePlot[SSRAlist, name, yearInc,
   opts] plots the occurrence rate of the named SSRA from a day
   before it begins to a day after it ends. OccurrenceRatePlot[
   SSRAlist, name, {startDate, endDate}, yearInc, opts] plots the
   occurrence rate of the named SSRA from startDate to endDate.
   OccurrenceRatePlot[SSRAlist, nameList, yearInc, opts] plots the
   occurrence rates of only SSRAs in nameList from a day before the
   earliest begins to a day after the last ends. OccurrenceRatePlot[
   SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots
   the occurrence rates of only SSRAs in nameList from startDate to
   endDate. opts is an optional argument for specifying Plot options."

AggregateOccurrenceRatePlot::"usage" =
 "AggregateOccurrenceRatePlot[SSRAlist, yearInc, opts] plots the
   aggregated occurrence rate (accidents per day) of all SSRAs in
   SSRAlist from a day before the earliest begins to a day after
   the last ends. yearInc specifies the calendar-year increments
   to be used on the plot. AggregateOccurrenceRatePlot[SSRAlist,
   {startDate, endDate}, yearInc, opts] plots the aggregated
   occurrence rate of all SSRAs from startDate to endDate.
   AggregateOccurrenceRatePlot[SSRAlist, nameList, yearInc, opts]
   plots the aggregated occurrence rate of only SSRAs in nameList
   from a day before the earliest begins to a day after the last
   ends. AggregateOccurrenceRatePlot[SSRAlist, nameList, {startDate,
   endDate}, yearInc, opts] plots the aggregated occurrence
   rate of only SSRAs in nameList from startDate to endDate.
   opts is an optional argument for specifying Plot options."

ExpectedAccidents::"usage" =
 "ExpectedAccidents[SSRAlist, name, {startDate, endDate}]
   calculates the expected number of accidents from
   startDate to endDate of the named SSRA from SSRAlist."
```

```
ExpectedAccidentsTable::"usage" =
 "ExpectedAccidentsTable[SSRAlist, startDate, year, opts]
   generates a table of n-year expected accidents for all SSRAs
   in SSRAlist from startDate forward. ExpectedAccidentsTable[
   SSRAlist, nameList, startDate, year, opts] generates a table
   of n-year expected accidents for only SSRAs in nameList from
   startDate forward. ExpectedAccidentsTable[SSRAlist, startDate,
   yearList, opts] generates a table for all SSRAs with multiple
   expected-accident columns where the years are specified
   by yearList. ExpectedAccidentsTable[SSRAlist, nameList,
   startDate, yearList, opts] generates a table with multiple
   expected-accident columns for only SSRAs in nameList. opts
   is an optional argument for specifying TableForm options."

ExpectedAccidentsPie::"usage" =
 "ExpectedAccidentsPie[SSRAlist, {startDate, endDate}, topSSRAs,
   opts] generates a pie chart of the expected number of accidents
   from startDate to endDate for the SSRAs in SSRAlist. Only the
   largest SSRAs, as specified by topSSRAs, are given their own
   wedges. The remaining SSRAs are placed in a single wedge.
   opts is an optional argument for specifying PieChart options."

ExpectedAccidentsPlot::"usage" =
 "ExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots the expected
   number of accidents for all SSRAs in SSRAlist from a day before
   the earliest begins to a day after the last ends. yearInc
   specifies the calendar-year increments to be used on the plot.
   ExpectedAccidentsPlot[SSRAlist, {startDate, endDate}, yearInc,
   opts] plots the expected number of accidents for all SSRAs from
   startDate to endDate. ExpectedAccidentsPlot[SSRAlist, name,
   yearInc, opts] plots the expected number of accidents for the
   named SSRA from a day before it begins to a day after it ends.
   ExpectedAccidentsPlot[SSRAlist, name, {startDate, endDate},
   yearInc, opts] plots the expected number of accidents for the named
   SSRA from startDate to endDate. ExpectedAccidentsPlot[SSRAlist,
   nameList, yearInc, opts] plots the expected number of accidents
   for only SSRAs in nameList from a day before the earliest begins
   to a day after the last ends. ExpectedAccidentsPlot[SSRAlist,
   nameList, {startDate, endDate}, yearInc, opts] plots the expected
   number of accidents for only SSRAs in nameList from startDate to
   endDate. opts is an optional argument for specifying Plot options."
```

```
AggregateExpectedAccidents::"usage" =
 "AggregateExpectedAccidents[SSRAlist, {startDate, endDate}]
   aggregates all SSRAs in SSRAlist from startDate to
   endDate. AggregateExpectedAccidents[SSRAlist, nameList,
   {startDate, endDate}] aggregates only SSRAs in nameList
   from startDate to endDate. AggregateExpectedAccidents[
   SSRAlist, startDate, years] aggregates all SSRAs in SSRAlist
   from startDate forward the specified number of years.
   AggregateExpectedAccidents[SSRAlist, nameList, startDate,
   years] aggregates only SSRAs in nameList from startDate forward."

AggregateExpectedAccidentsPlot::"usage" =
 "AggregateExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots
   the aggregate of all SSRAs in SSRAlist from a day before
   the earliest begins to a day after the last ends. yearInc
   specifies the calendar-year increments to be used on the
   plot. AggregateExpectedAccidentsPlot[SSRAlist, {startDate,
   endDate}, yearInc, opts] plots the aggregate of all SSRAs from
   startDate to endDate. AggregateExpectedAccidentsPlot[SSRAlist,
   nameList, yearInc, opts] plots the aggregate of only SSRAs
   in nameList from a day before the earliest begins to a day
   after the last ends. AggregateExpectedAccidentsPlot[SSRAlist,
   nameList, {startDate, endDate}, yearInc, opts] plots the
   aggregate of only SSRAs in nameList from startDate to endDate.
   opts is an optional argument for specifying Plot options."

ExpectedAccidentsTrendPlot::"usage" =
 "ExpectedAccidentsTrendPlot[SSRAlist, endDate, year, yearInc,
   opts] plots the n-year aggregate expected-accident trend for
   all SSRAs in SSRAlist from the beginning of the earliest SSRA
   until endDate. yearInc specifies the calendar-year increments
   to be used on the plot. ExpectedAccidentsTrendPlot[SSRAlist,
   {startDate, endDate}, year, yearInc, opts] plots the n-year
   expected accident trend for all SSRAs from startDate to endDate.
   opts is an optional argument for specifying Plot options."

FlightHoursTest::"usage" =
 "FlightHoursTest[FHdata] tests a flight-hours
   list to ensure it's structured correctly."

FlightHours::"usage" =
 "FlightHours[FHdata, {startDate, endDate}] calculates
   the number of flight hours accrued between startDate
   and endDate given the flight-hours data in FHdata."
```

```
FlightHoursDate::"usage" =
 "FlightHoursDate[FHdata, startDate, fhrs}] calculates,
   starting from startDate, the date when fhrs flight hours
   will be accumulated, given the flight-hours data in FHdata."


ExpectedAccidentsNext::"usage" =
 "ExpectedAccidentsNext[SSRAlist, FHdata, startDate, fhrs] calculates,
   starting from startDate, the aggregate number of accidents
   expected to occur in the next fhrs flight hours, given the
   SSRA data in SSRAlist and the flight-hours data in FHdata."


ExpectedAccidentsNextPlot::"usage" =
 "ExpectedAccidentsNextPlot[SSRAlist, FHdata, {startDate,
   endDate}, fhrs, yearInc, opts] plots, between startDate
   and endDate, the aggregate number of accidents expected to
   occur in the next fhrs flight hours, given the SSRA data
   in SSRAlist and the flight-hours data in FHdata. yearInc
   specifies the calendar-year increments to be used on the plot.
   opts is an optional argument for specifying Plot options."
```

■ **Error messages for the exported objects**

Error messages for SSRATest:

```
SSRATest::"ssralength" = "SSRA length must be an even integer ≥ 6."


SSRATest::"ssraname" =
 "First element of SSRA must be the SSRA name (string)."


SSRATest::"ssramodel" = "Second element of SSRA must
   be an indicator of helicopter applicability (string)."


SSRATest::"ssraseverity" =
 "Third element of SSRA must be I, II, III or IV (string)."


SSRATest::"ssraexpectedaccidents" =
 "Odd-numbered elements 5 and up, expected-
   accident predictions, must be non-negative numbers."


SSRATest::"ssradatelength" = "Even-numbered elements
   4 and up, dates, must be a list with three elements."
```

```
SSRATest::"ssradateyear" =
 "The first sub-element of the even-numbered elements
   4 and up must an integer between 1900 and 2100 (year)."


SSRATest::"ssradatemonth" =
 "The second sub-element of the even-numbered elements
   4 and up must an integer between 1 and 12 (month)."


SSRATest::"ssradateday" =
 "The third sub-element of the even-numbered elements
   4 and up must an integer between 1 and 31 (day)."


SSRATest::"ssradatechrono" =
 "The even-numbered elements 4 and up must be
   chronologically-ordered dates in the form {year, month, day}."


SSRATest::"ssradateequal" = "Dates must not be equal."
```

Error messages for `FlightHoursTest`:

```
FlightHoursTest::"fhourslength" =
 "Length of flight-hours list must be an odd integer ≥ 3."


FlightHoursTest::"fhoursnonnegnumbers" =
 "Even numbered elements, flight-hour
   predictions, must be non-negative numbers."


FlightHoursTest::"fhoursdatelength" =
 "Odd-numbered elements, dates, must be a list with three elements."


FlightHoursTest::"fhoursdateyear" =
 "The first sub-element of the odd-numbered
   elements must an integer between 1900 and 2100 (year)."


FlightHoursTest::"fhoursdatemonth" =
 "The second sub-element of the odd-numbered
   elements must an integer between 1 and 12 (month)."


FlightHoursTest::"fhoursdateday" =
 "The third sub-element of the odd-numbered
   elements must an integer between 1 and 31 (day)."
```

```
FlightHoursTest::"fhoursdatechrono" =
 "The odd-numbered elements must be chronologically-
   ordered dates in the form {year, month, day}."

FlightHoursTest::"fhoursdateequal" = "Dates must not be equal."
```

Implementation

This part contains the actual definitions and any auxiliary functions that should not be visible outside.

■ **Begin the private context (implementation part)**

```
Begin["`Private`"]
```

■ **Read in any hidden imports**

None.

■ **Unprotect any system functions for which definitions will be made**

None.

■ **Definition of auxiliary functions and local (static) variables**

dateconv[]

It will be useful to define an auxiliary function `dateconv` that uses the built-in function `FromDate` to convert dates from {year, month, day} format to an absolute number of days since 1 Jan 1900.

```
dateconv[ymd_] := FromDate[Flatten[{ymd, 0, 0, 0}]] / (60 60 24)
```

occRate[]

`occRate` is an auxiliary function for occurrence-rate calculations.

```
occRate[ssra_, day_] := Which[Evaluate[
   Sequence @@ Flatten[({dateconv[ssra[[#1]]] ≤ day < dateconv[ssra[[#1 + 2]]],
        ssra[[#1 + 1]] / DaysBetween[ssra[[#1]], ssra[[#1 + 2]]]} &) /@
     Range[4, Length[ssra] - 2, 2]]], True, 0]
```

expAcc[]

expAcc is an auxiliary function for expected-accident calculations. It works by first calculating the number of days between the arguments *startday* and *endDay* (in days after 1 Jan 1900) for each expected-accident prediction provided in the *ssra.* The resulting list is assigned as the value of the local variable *rawqty.* Any negative values found in *rawqty* are then replaced by zeros and the result is assigned as the value of *qty.* A negative value occurs when a prediction doesn't fall between *startDay* and *endDay.* Then a list of occurrence rates (in accidents per day) is generated from each of the expected-accident predictions in *ssra* and assigned as the value of *ORates.* Each quantity of days in *qty* is multiplied by the corresponding occurrence rate in *ORates* which results in a list of expected-accident predictions prorated by *startDay* and *endDay.* Finally the elements of this list of predictions are summed yielding the expected number of accidents between *startDay* and *endDay* for the specified SSRA.

```
expAcc[ssra_, startDay_, endDay_] := Module[
  {rawqty, qty, ORates}, rawqty = (Min[dateconv[ssra[[#1 + 2]]], endDay] -
      Max[dateconv[ssra[[#1]]], startDay] &) /@
   Range[4, Length[ssra] - 2, 2]; qty = (If[#1 ≥ 0, #1, 0] &) /@ rawqty;
  ORates = (ssra[[#1]] / (dateconv[ssra[[#1 + 1]]] - dateconv[ssra[[#1 - 1]]]) &) /@
   Range[5, Length[ssra] - 1, 2]; Plus @@ (qty ORates)]
```

yearticks[]

yearticksc is an auxiliary function for generating calendar-ticks.

```
yearticks[pMin_, pMax_, yInc_] :=
 ({dateconv[{#1, 1, 1}], ToString[#1]} &) /@
  Range[First[ToDate[pMin 60^2 24]] + 1, First[ToDate[pMax 60^2 24]], yInc]
```

fHours[]

fHours is an auxiliary function for calculating flight hours. It calculates the quantity of flight hours between *startDay* and *endDay* given the flight-hour data in *FHdata.*

```
fHours[FHdata_, startDay_, endDay_] := Module[
  {rawqty, qty, FHrates}, rawqty = (Min[dateconv[FHdata[[#+2]]], endDay] -
      Max[dateconv[FHdata[[#]]], startDay] &) /@
   Range[1, Length[FHdata] - 2, 2]; qty = (If[# ≥ 0, #, 0] &) /@ rawqty;
  FHrates = (FHdata[[#]] / (dateconv[FHdata[[#+1]]] - dateconv[FHdata[[#-1]]]) &) /@
   Range[2, Length[FHdata] - 1, 2]; Plus @@ (qty FHrates)]
```

FHdate[]

FHdate is an auxiliary function that calculates the number of days after *startDay* when a specified number of flight hours will be accumulated. The argument *FHdata* refers to the flight-hours data structure and the argument *fhrs* specifies the quantity of flight hours. *startDay* and the result that FHdate returns are in days after 1 Jan 1900. This function works by first setting the local variable *end* equal to *startDay*, then using a While loop to increment *end* one day at a time until the number of flight hours exceeds the specified threshold. The value of *end* that resulted in the flight-hour threshold being breached is then returned.

```
FHdate[FHdata_, startDay_, fhrs_] := Module[{end}, end = startDay;
  While[fHours[FHdata, startDay, end] < fhrs, end++]; end]
```

expaccnext[]

expaccnext is an auxiliary function that calculates, starting from *startDay*, the aggregate number of accidents expected to occur in the next *fhrs* flight hours. The arguments *allSSRAs* and *FHdata* refer to the SSRA and the flight-hours data structures, respectively. The argument *fhrs* specifies the quantity of flight hours. This function relies on the utility function FHdate to calculate the date at which the specified quantity of flight hours will be reached and then calculates the corresponding aggregate number of accidents expected.

```
expaccnext[allSSRAs_, FHdata_, startDay_, fhrs_] :=
 Module[{endDay}, endDay = FHdate[FHdata, startDay, fhrs];
  Plus @@ (expAcc[#1, startDay, endDay] &) /@ allSSRAs]
```

- **Set options**

Default values for the options of Plot are set here:

```
SetOptions[Plot, PlotRange → All, Frame → True,
 Axes → False, PlotStyle → Thickness[0.01]]
```

■ **Definition of the exported functions**

SSRATest[]

This function tests an SSRA to ensure it is structured correctly. *test1* ensures that the length is an even integer greater than or equal to 6. *test2* and *test3* ensure the first two elements are strings. *test4* ensures the third element is "I", "II", "III" or "IV". *test5* determines whether the odd elements 5 and up are non-negative numbers. This doesn't ensure that these numbers aren't Complex. *test6* determines whether the even elements 4 and up are vectors of length 3. *test7* determines whether the first sub-element of even elements 4 and up are integers between 1900 and 2100. *test8* determines whether the second sub-element of even elements 4 and up are integers between 1 and 12. *test9* determines whether the third sub-element of even elements 4 and up are integers between 1 and 31. This still doesn't completely guarantee a real date since it is possible that one might input Feb 29 (non leap year), 30 or 31, Apr 31, Jun 31, Sep 31 or Nov 31. *test10* determines whether even elements 4 and up are chronologically-ordered but doesn't flag situations where two adjacent dates are equal. *test11* determines whether two adjacent dates are equal.

```
SSRATest[ssra_List] :=
 Module[{len, test1, test2, test3, test4, test5, test6, test7, test8,
   test9, test10}, len = Length[ssra]; test1 = EvenQ[len] && len ≥ 6;
  test2 = Head[ssra[[1]]] === String; test3 = Head[ssra[[2]]] === String;
  test4 = (#1 === "I" || #1 === "II" || #1 === "III" || #1 === "IV" &) [ssra[[3]]];
  test5 = And @@
    (NumberQ[ssra[[#1]]] && NonNegative[ssra[[#1]]] &) /@ Range[5, len, 2];
  test6 = And @@ (VectorQ[ssra[[#1]]] && Length[ssra[[#1]]] == 3 &) /@
     Range[4, len, 2]; test7 = And @@
    (IntegerQ[ssra[[#1, 1]]] && 1900 < ssra[[#1, 1]] < 2100 &) /@ Range[4, len, 2];
  test8 = And @@ (IntegerQ[ssra[[#1, 2]]] && 1 ≤ ssra[[#1, 2]] ≤ 12 &) /@
     Range[4, len, 2]; test9 =
   And @@ (IntegerQ[ssra[[#1, 3]]] && 1 ≤ ssra[[#1, 3]] ≤ 31 &) /@ Range[4, len, 2];
  test10 = And @@ (OrderedQ[ssra[[#1]] < ssra[[#1 + 2]]] &) /@ Range[4, len - 2, 2];
  test11 = And @@ (ssra[[#1]] =!= ssra[[#1 + 2]] &) /@ Range[4, len - 2, 2];
  Which[! test1, Message[SSRATest::"ssralength"],
   ! test2, Message[SSRATest::"ssraname"],
   ! test3, Message[SSRATest::"ssramodel"], ! test4,
   Message[SSRATest::"ssraseverity"], ! test5,
   Message[SSRATest::"ssraexpectedaccidents"],
   ! test6, Message[SSRATest::"ssradatelength"],
   ! test7, Message[SSRATest::"ssradateyear"],
   ! test8, Message[SSRATest::"ssradatemonth"],
   ! test9, Message[SSRATest::"ssradateday"], ! test10,
   Message[SSRATest::"ssradatechrono"], ! test11,
   Message[SSRATest::"ssradateequal"], 1 == 1, True]]
```

SSRASelect[]

SSRASelect selects SSRAs that have a specific severity level from an SSRA list that is mixed. This is done by selecting all elements in the mixed list which have either "I", "II", "III" or "IV" as their third element.

In order to avoid the accumulation of round-off errors during analysis, the expected accident quantities, most of which are expressed as real numbers, will be rationalized with the built-in function Rationalize as recommended by McCullough (2000):

```
SSRASelect[SSRAList_List, sevlev_String] :=
 Select[SSRAList, #1[[3]] == sevlev &] /. n_Real :> Rationalize[n, 0]
```

FirstSSRA[]

FirstSSRA selects from a list of SSRAs the SSRA that began first.

```
FirstSSRA[SSRAlist_List] :=
 First[Sort[SSRAlist, OrderedQ[{#1[[4]], #2[[4]]}] &]]
```

LastSSRA[]

LastSSRA selects from a list of SSRAs the SSRA that began last.

```
LastSSRA[SSRAlist_List] := Last[Sort[SSRAlist, OrderedQ[{#1[[4]], #2[[4]]}] &]]
```

OccurrenceRate[]

We will assume that, for any component SSRA, the number of accidents across the fleet (or sub-fleet) during a year follows the Poisson distribution. The yearly expected accident quantity for each SSRA is then the parameter of the Poisson distribution. This parameter is the product of the occurrence rate and the exposure time. Here is the definition of the function OccurrenceRate which calculates the occurrence rate, in accidents per day, for any SSRA in the list *modelsev*.

```
OccurrenceRate[modelsev_List, name_String,
  date_ /; VectorQ[date] && Length[date] == 3] :=
 Module[{days, SSRA}, days = dateconv[date];
  SSRA = Flatten[Select[modelsev, #1[[1]] == name &], 1]; occRate[SSRA, days]]
```

```
OccurrenceRate[modelsev_List, name_String, date_Symbol] :=
 Module[{days, SSRA}, days = Hold[FromDate[Flatten[{date, 0, 0, 0}]] / (60 60 24)];
  SSRA = Flatten[Select[modelsev, #1[[1]] == name &], 1];
  Which[Evaluate[Sequence @@
     Flatten[({dateconv[SSRA[[#1]]] ≤ days < dateconv[SSRA[[#1 + 2]]],
          SSRA[[#1 + 1]] / DaysBetween[SSRA[[#1]], SSRA[[#1 + 2]]]} &) /@
       Range[4, Length[SSRA] - 2, 2]]], True, 0]]
```

The occurrence rate uses the built-in function `Which` to test *days* and assign the appropriate accidents-per-day value. The argument *date* must be in format {year, month, day}.

OccurrenceRateTable[]

The version of `OccurrenceRateTable` below generates a table of occurrence rates in accidents per day on the specified date for the selected SSRAs. This pattern matches when the first argument is a list, the second is a vector of strings and the third is a vector of length 3.

```
OccurrenceRateTable[modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  date_ /; VectorQ[date] && Length[date] == 3, opts___?OptionQ] :=
 Module[{SSRAlist, occRateList, occRateSum, perList},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  occRateList = (occRate[#1, dateconv[date]] &) /@ SSRAlist;
  occRateSum = Plus @@ occRateList;
  perList = (100 occRateList) / occRateSum; TableForm[Append[
    Sort[Transpose[{nameList, occRateList, perList}], #1[[2]] > #2[[2]] &],
    {"SUM", occRateSum, Plus @@ perList}], opts, TableHeadings →
    {None, {"SSRA", "OCCURRENCE RATE (ACCIDENTS/DAY)", "PERCENT"}}]]
```

The version of `OccurrenceRateTable` below generates a table of occurrence rates in accidents per day on the specified date for all the SSRAs in *modelsev*. This pattern matches when the first argument is a list and the second is a vector of length 3.

```
OccurrenceRateTable[modelsev_List,
  date_ /; VectorQ[date] && Length[date] == 3, opts___?OptionQ] :=
 Module[{names, occRateList, occRateSum, perList},
  names = (First[#1] &) /@modelsev;
  occRateList = (occRate[#1, dateconv[date]] &) /@modelsev;
                                            100 occRateList
  occRateSum = Plus @@ occRateList; perList = ---------------; TableForm[
                                              occRateSum
   Append[Sort[Transpose[{names, occRateList, perList}], #1〚2〛 > #2〚2〛 &],
    {"SUM", occRateSum, Plus @@ perList}], opts, TableHeadings →
    {None, {"SSRA", "OCCURRENCE RATE (ACCIDENTS/DAY)", "PERCENT"}}]]
```

opts is an optional argument for specifying `TableForm` options.

OccurrenceRatePie[]

`OccurrenceRatePie` generates a pie chart of occurrence rates in accidents per day on the specified date for the SSRAs in modelsev. Only the largest SSRAs, as specified by *topSSRAs*, are given their own wedges. The remaining SSRAs are placed in a single wedge.

This function constructs a list of occurrence rates and names for each of the SSRAs, sorts them from largest to smallest, then transposes the list so that the sorted occurrence rates and names are in the first and second sublists, respectively, of the local variable *occnames*. The occurrence rates not included in the *topSSRAs* are summed, this sum takes the *topSSRAs* + 1 place in the sorted occurrence rates and result is assigned as the value of *pievalues*. The sorted names not included in the *topSSRAs* are replaced by the name "_ SSRAs" where the blank is replaced by the quantity of SSRAs being lumped together. This revised name list is assigned as the value of *pielabels*. Finally, the function `PieChart` from the standard add-on package *Graphics`Graphics`* generates the pie chart using *pievalues* and *pielabels*. *opts* is an optional argument for specifying `PieChart` options.

```
OccurrenceRatePie[modelsev_List,
  date_ /; VectorQ[date] && Length[date] == 3,
  topssras_Integer, opts___?OptionQ] :=
 Module[{occnames, pievalues, pielabels}, occnames = Transpose[Sort[
     Map[{occRate[#, dateconv[date]], #〚1〛} &, modelsev], #1〚1〛 > #2〚1〛 &]];
  pievalues = Take[ReplacePart[occnames〚1〛, Apply[Plus, Take[occnames〚1〛,
       topssras - Length[occnames〚1〛]]], topssras + 1], topssras + 1];
  pielabels = Append[Take[occnames〚2〛, topssras],
    StringForm["`1` SSRAs", Length[occnames〚2〛] - topssras]];
  PieChart[pievalues, opts, PieLabels → pielabels,
   PlotLabel → "Occurrence Rates"]]
```

OccurrenceRatePlot[]

The version of OccurrenceRatePlot below plots the occurrence rate (accidents per day) of a single SSRA from a day before the SSRA begins to a day after it ends. This pattern will match when the first argument supplied is a list, the second is a string and the third is an integer.

```
OccurrenceRatePlot[modelsev_List, name_String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRA, plotMin, plotMax},
  SSRA = Flatten[Select[modelsev, #1[[1]] == name &], 1];
  plotMin = dateconv[SSRA[[4]]] - 1; plotMax = dateconv[Last[SSRA]] + 1;
  Plot[Evaluate[occRate[SSRA, days]],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", name, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of OccurrenceRatePlot below plots the occurrence rate (accidents per day) of a single SSRA from a specified start date to a specified ending date. This pattern will match when the first argument supplied is a list, the second is a string, the third is a list containing two vectors of length three and the fourth is an integer.

```
OccurrenceRatePlot[modelsev_List, name_String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{SSRA, plotMin, plotMax},
  SSRA = Flatten[Select[modelsev, #1[[1]] == name &], 1];
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[occRate[SSRA, days]], {days, plotMin, plotMax},
   opts, PlotRange → {All, {0, Automatic}}, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", name, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of OccurrenceRatePlot below plots the occurrence rates (accidents per day) of multiple SSRAs from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list, the second is a list of strings and the third is an integer.

```
OccurrenceRatePlot[modelsev_List,
  nameList_ /; VectorQ[nameList] && SameQ @@ Head /@ nameList &&
    Head[nameList[[1]]] == String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = Min[(dateconv[#1[[4]]] &) /@ SSRAlist] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ SSRAlist] + 1;
  Plot[Evaluate[(occRate[#1, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `OccurrenceRatePlot` below plots the occurrence rates (accidents per day) of multiple SSRAs from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list of strings, the third is a list containing two vectors of length three and the fourth is an integer.

```
OccurrenceRatePlot[modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[(occRate[#1, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `OccurrenceRatePlot` below plots the occurrence rates (accidents per day) of all the SSRAs in *modelsev* from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list and the second is an integer.

```
OccurrenceRatePlot[modelsev_List, yearInc_Integer, opts___?OptionQ] :=
 Module[{plotMin, plotMax},
  plotMin = Min[(dateconv[#1[[4]]] &) /@ modelsev] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ modelsev] + 1;
  Plot[Evaluate[(occRate[#1, days] &) /@ modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `OccurrenceRatePlot` below plots the occurrence rates (accidents per day) of all the SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list containing two vectors of length three and the third is an integer.

```
OccurrenceRatePlot[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[(occRate[#1, days] &) /@modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Occur. Rate (accidents/day)", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

opts is an optional argument for specifying `Plot` options.

AggregateOccurrenceRatePlot[]

The version of `AggregateOccurrenceRatePlot` below plots the aggregate occurrence rate (accidents per day) of multiple SSRAs from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list, the second is a list of strings and the third is an integer.

```
AggregateOccurrenceRatePlot[modelsev_List,
  nameList_ /; VectorQ[nameList] && SameQ @@ Head /@ nameList &&
    Head[nameList[[1]]] == String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@modelsev;
  plotMin = Min[(dateconv[#1[[4]]] &) /@ SSRAlist] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ SSRAlist] + 1;
  Plot[Evaluate[Plus @@ (occRate[#1, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts,
   PlotRange → {All, {0, Automatic}}, FrameLabel → {"Calendar Year",
     "Occur. Rate (accidents/day)", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateOccurrenceRatePlot` below plots the aggregate occurrence rate (accidents per day) of multiple SSRAs from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list of strings, the third is a list containing two vectors of length three and the fourth is an integer.

```
AggregateOccurrenceRatePlot[
  modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[Plus @@ (occRate[#1, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Occur. Rate (accidents/day)", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateOccurrenceRatePlot` below plots the aggregate occurrence rate (accidents per day) of all the SSRAs in *modelsev* from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list and the second is an integer.

```
AggregateOccurrenceRatePlot[modelsev_List,
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = Min[(dateconv[#1[[4]]] &) /@ modelsev] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ modelsev] + 1;
  Plot[Evaluate[Plus @@ (occRate[#1, days] &) /@ modelsev],
   {days, plotMin, plotMax}, opts,
   PlotRange → {All, {0, Automatic}}, FrameLabel → {"Calendar Year",
     "Occur. Rate (accidents/day)", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateOccurrenceRatePlot` below plots the aggregate occurrence rate (accidents per day) of all the SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list containing two vectors of length three and the third is an integer.

```
AggregateOccurrenceRatePlot[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[Plus @@ (occRate[#1, days] &) /@ modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Occur. Rate (accidents/day)", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

opts is an optional argument for specifying `Plot` options.

ExpectedAccidents[]

ExpectedAccidents calculates the number of accidents expected to occur for the named SSRA between *startDate* and *endDate*. This function relies on the auxiliary function expacc. Refer to the description of that function for further information on how it works.

```
ExpectedAccidents[modelsev_List, name_String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}] :=
 Module[{SSRA}, SSRA = Flatten[Select[modelsev, #1[[1]] == name &], 1];
  expAcc[SSRA, dateconv[startDate], dateconv[endDate]]]
```

ExpectedAccidentsTable[]

ExpectedAccidentsTable generates an expected-accidents table with multiple columns as specified by *yearList* for only the SSRAs in *nameList*. It matches when the first argument is a list, the second is a vector of strings, the third is a vector of length 3 and the fourth is a vector of integers.

```
ExpectedAccidentsTable[modelsev_List, nameList_ /;
    VectorQ[nameList] && SameQ @@ Head /@ nameList &&
     Head[nameList[[1]]] == String, startDate_ /;
    VectorQ[startDate] && Length[startDate] == 3,
   yearList_ /; VectorQ[yearList] && SameQ @@ Head /@ yearList &&
     Head[yearList[[1]]] == Integer, (opts___)?OptionQ] :=
  Module[{SSRAlist, endDateList, expAccList},
   SSRAlist = nameList /. (#1[[1]] -> #1 & ) /@ modelsev;
    endDateList = (ReplacePart[startDate, startDate[[1]] + #1, 1] & )
/@
      yearList; expAccList =
     ((Function[ssra, expAcc[ssra, dateconv[startDate], dateconv[#1]]])
/@
        SSRAlist & ) /@ endDateList;
    TableForm[Append[Sort[Transpose[Prepend[expAccList, nameList]],
       #1[[2]] > #2[[2]] & ], Prepend[(Plus @@ #1 & ) /@ expAccList,
"SUM"]],
     opts, TableHeadings -> {None, Prepend[
        (ToString[StringForm["EXP. ACC. `1` YEAR(s)", #1]] & ) /@
yearList,
        "SSRA"]}]]
```

ExpectedAccidentsTable generates an expected-accidents table with multiple columns as specified by *yearList* for all the SSRAs. It matches when the first argument is a list, the second is a vector of length 3 and the third is a vector of integers.

```
ExpectedAccidentsTable[modelsev_List, startDate_ /;
    VectorQ[startDate] && Length[startDate] == 3,
   yearList_ /; VectorQ[yearList] && SameQ @@ Head /@ yearList &&
     Head[yearList[[1]]] == Integer, (opts___)?OptionQ] :=
  Module[{names, endDateList, expAccList},
   names = (First[#1] & ) /@ modelsev; endDateList =
     (ReplacePart[startDate, startDate[[1]] + #1, 1] & ) /@ yearList;
    expAccList = ((Function[ssra, expAcc[ssra, dateconv[startDate],
          dateconv[#1]]]) /@ modelsev & ) /@ endDateList;
    TableForm[Append[Sort[Transpose[Prepend[expAccList, names]],
       #1[[2]] > #2[[2]] & ], Prepend[(Plus @@ #1 & ) /@ expAccList,
"SUM"]],
     opts, TableHeadings -> {None, Prepend[
        (ToString[StringForm["EXP. ACC. `1` YEAR(s)", #1]] & ) /@
yearList,
       "SSRA"]}]]
```

ExpectedAccidentsTable generates a table of n-year expected accidents for the SSRAs specified in *nameList* from *startDate* forward. A column of percentages is included. It matches when the first argument is a list, the second is a vector of strings, the third is a vector of length 3 and the fourth is an integer.

```
ExpectedAccidentsTable[modelsev_List, nameList_ /;
    VectorQ[nameList] && SameQ @@ Head /@ nameList &&
     Head[nameList[[1]]] == String, startDate_ /;
    VectorQ[startDate] && Length[startDate] == 3, year_Integer,
   (opts___)?OptionQ] := Module[{SSRAlist, endDate, expAccList,
expPerList,
    expSum}, SSRAlist = nameList /. (#1[[1]] -> #1 & ) /@ modelsev;
    endDate = ReplacePart[startDate, startDate[[1]] + year, 1];
    expAccList = (expAcc[#1, dateconv[startDate], dateconv[endDate]]
& ) /@
      SSRAlist; expPerList = (expAccList*100)/expSum;
    expSum = Plus @@ expAccList; TableForm[
     Append[Sort[Transpose[{nameList, expAccList, expPerList}],
       #1[[2]] > #2[[2]] & ], {"Sum", expSum, Plus @@ expPerList}],
opts,
     TableHeadings -> {None, {"SSRA", ToString[
         StringForm["EXP. ACC. `1` YEAR(s)", year]], "PERCENT"}}]]
```

ExpectedAccidentsTable generates a table of n-year expected accidents for all the SSRAs from *startDate* forward. A column of percentages is included. It matches when the first argument is a list, the second is a vector of length 3 and the third is an integer.

```
ExpectedAccidentsTable[modelsev_List, startDate_ /;
    VectorQ[startDate] && Length[startDate] == 3, year_Integer,
   (opts___)?OptionQ] := Module[{names, endDate, expAccList,
expPerList,
    expSum}, names = (First[#1] & ) /@ modelsev;
    endDate = ReplacePart[startDate, startDate[[1]] + year, 1];
    expAccList = (expAcc[#1, dateconv[startDate], dateconv[endDate]]
& ) /@
      modelsev; expPerList = (expAccList*100)/expSum;
    expSum = Plus @@ expAccList; TableForm[
     Append[Sort[Transpose[{names, expAccList, expPerList}],
       #1[[2]] > #2[[2]] & ], {"Sum", expSum, Plus @@ expPerList}],
opts,
     TableHeadings -> {None, {"SSRA", ToString[
         StringForm["EXP. ACC. `1` YEAR(s)", year]], "PERCENT"}}]]
```

opts is an optional argument for specifying `TableForm` options.

ExpectedAccidentsPie[]

`ExpectedAccidentsPie` generates a pie chart of the expected number of accidents from *startDate* to *endDate* for the SSRAs in *modelsev*. Only the largest SSRAs, as specified by *topSSRAs*, are given their own wedges. The remaining SSRAs are placed in a single wedge.

This function constructs a list of expected-accident values and names for each of the SSRAs, sorts them from largest to smallest, then transposes the list so that the sorted expected-accident values and names are in the first and second sublists, respectively, of the local variable *eanames*. The expected-accident values not included in the *topSSRAs* are summed, this sum takes the *topSSRAs* + 1 place in the sorted expected-accident values and result is assigned as the value of *pievalues*. The sorted names not included in the *topSSRAs* are replaced by the name "_ SSRAs" where the blank is replaced by the quantity of SSRAs being lumped together. This revised name list is assigned as the value of *pielabels*. Finally, the function `PieChart` from the standard add-on package *Graphics`Graphics`* generates the pie chart using *pievalues* and *pielabels*. *opts* is an optional argument for specifying `PieChart` options.

```
ExpectedAccidentsPie[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  topssras_Integer, opts___?OptionQ] :=
 Module[{eanames, pievalues, pielabels}, eanames = Transpose[
    Sort[Map[{expAcc[#, dateconv[startDate], dateconv[endDate]], #⟦1⟧} &,
      modelsev], #1⟦1⟧ > #2⟦1⟧ &]];
  pievalues = Take[ReplacePart[eanames⟦1⟧, Apply[Plus, Take[eanames⟦1⟧,
       topssras - Length[eanames⟦1⟧]]], topssras + 1], topssras + 1];
  pielabels = Append[Take[eanames⟦2⟧, topssras],
    StringForm["`1` SSRAs", Length[eanames⟦2⟧] - topssras]];
  PieChart[pievalues, opts, PieLabels → pielabels,
   PlotLabel → "Expected Accidents"]]
```

ExpectedAccidentsPlot[]

The version of `ExpectedAccidentsPlot` below plots the expected number of accidents for a single SSRA from a day before the SSRA begins to a day after it ends. This version will apply when the first argument supplied is a list, the second is a string and the third is an integer.

```
ExpectedAccidentsPlot[modelsev_List, name_String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRA, plotMin, plotMax},
  SSRA = Flatten[Select[modelsev, #1⟦1⟧ == name &], 1];
  plotMin = dateconv[SSRA⟦4⟧] - 1; plotMax = dateconv[Last[SSRA]] + 1;
  Plot[Evaluate[expAcc[SSRA, plotMin, days]],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", name, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `ExpectedAccidentsPlot` below plots the expected number of accidents for a single SSRA from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a string, the third is a list containing two vectors of length three and the fourth is an integer.

```
ExpectedAccidentsPlot[modelsev_List, name_String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{SSRA, plotMin, plotMax},
  SSRA = Flatten[Select[modelsev, #1⟦1⟧ == name &], 1];
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[expAcc[SSRA, dateconv[startDate], days]], {days, plotMin,
    plotMax}, opts, PlotRange → {All, {0, Automatic}}, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", name, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of ExpectedAccidentsPlot below plots the expected number of accidents for multiple SSRAs from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list, the second is a list of strings and the third is an integer.

```
ExpectedAccidentsPlot[modelsev_List,
  nameList_ /; VectorQ[nameList] && SameQ @@ Head /@ nameList &&
    Head[nameList[[1]]] == String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = Min[(dateconv[#1[[4]]] &) /@ SSRAlist] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ SSRAlist] + 1;
  Plot[Evaluate[(expAcc[#1, plotMin, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of ExpectedAccidentsPlot below plots the expected number of accidents for multiple SSRAs from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list of strings, the third is a list containing two vectors of length three and the fourth is an integer.

```
ExpectedAccidentsPlot[modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[(expAcc[#1, plotMin, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of ExpectedAccidentsPlot below plots the expected number of accidents for all the SSRAs in *modelsev* from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list and the second is an integer.

```
ExpectedAccidentsPlot[modelsev_List, yearInc_Integer,
  opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = Min[(dateconv[#1[[4]]] &) /@modelsev] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@modelsev] + 1;
  Plot[Evaluate[(expAcc[#1, plotMin, days] &) /@modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `ExpectedAccidentsPlot` below plots the expected number of accidents for all the SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list containing two vectors of length three and the third is an integer.

```
ExpectedAccidentsPlot[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[(expAcc[#1, plotMin, days] &) /@modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel →
    {"Calendar Year", "Cumulative Accidents Expected", None, None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

opts is an optional argument for specifying `Plot` options.

AggregateExpectedAccidents[]

This function calculates the expected number of accidents for each SSRA and then sums them. This is done by mapping the auxiliary function `expAcc` over the list of SSRAs, and then applying `Plus` to the result.

The version of `AggregateExpectedAccidents` below aggregates the expected number of accidents for all SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list and the second is a list containing two vectors of length three.

```
AggregateExpectedAccidents[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}] :=
 Plus @@ (expAcc[#1, dateconv[startDate], dateconv[endDate]] &) /@modelsev
```

The version of AggregateExpectedAccidents below aggregates the expected number of accidents for selected SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a vector of strings and the third is a list containing two vectors of length three.

```
AggregateExpectedAccidents[
  modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}] :=
 Module[{SSRAlist}, SSRAlist = nameList /. (#1[[1]] → #1 &) /@modelsev; Plus @@
   (expAcc[#1, dateconv[startDate], dateconv[endDate]] &) /@ SSRAlist]
```

The version of AggregateExpectedAccidents below aggregates the expected number of accidents for all the SSRAs in *modelsev* from a specified start date to a date one or more years in the future. This version will apply when the first argument supplied is a list, the second is a vector of length three and the third is an integer.

```
AggregateExpectedAccidents[modelsev_List,
  startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
  years_Integer] := Plus @@ (expAcc[#1, dateconv[startDate], dateconv[
       ReplacePart[startDate, startDate[[1]] + years, 1]]] &) /@modelsev
```

The version of AggregateExpectedAccidents below aggregates the expected number of accidents for selected SSRAs in *modelsev* from a specified start date to a date one or more years in the future. This version will apply when the first argument supplied is a list, the second is a vector of strings, the third is a vector of length three and the fourth is an integer.

```
AggregateExpectedAccidents[
  modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
  years_Integer] :=
 Module[{SSRAlist}, SSRAlist = nameList /. (#1[[1]] → #1 &) /@modelsev;
  Plus @@ (expAcc[#1, dateconv[startDate], dateconv[
        ReplacePart[startDate, startDate[[1]] + years, 1]]] &) /@ SSRAlist]
```

AggregateExpectedAccidentsPlot[]

The version of `AggregateExpectedAccidentsPlot` below plots the aggregate expected number of accidents of the SSRAs listed in *nameList* from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list, the second is a list of strings and the third is an integer.

```
AggregateExpectedAccidentsPlot[modelsev_List,
  nameList_ /; VectorQ[nameList] && SameQ @@ Head /@ nameList &&
    Head[nameList[[1]]] == String, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = Min[(dateconv[#1[[4]]] &) /@ SSRAlist] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@ SSRAlist] + 1;
  Plot[Evaluate[Plus @@ (expAcc[#1, plotMin, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Cumulative Accidents Expected", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateExpectedAccidentsPlot` below plots the aggregate expected number of accidents of the SSRAs listed in *nameList* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list of strings, the third is a list containing two vectors of length three and the fourth is an integer.

```
AggregateExpectedAccidentsPlot[
  modelsev_List, nameList_ /; VectorQ[nameList] &&
    SameQ @@ Head /@ nameList && Head[nameList[[1]]] == String,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}, yearInc_Integer,
  opts___?OptionQ] := Module[{SSRAlist, plotMin, plotMax},
  SSRAlist = nameList /. (#1[[1]] → #1 &) /@ modelsev;
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[Plus @@ (expAcc[#1, plotMin, days] &) /@ SSRAlist],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Cumulative Accidents Expected", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateExpectedAccidentsPlot` below plots the aggregate expected number of accidents of all the SSRAs in *modelsev* from a day before the earliest SSRA begins to a day after the last one ends. This version will apply when the first argument supplied is a list and the second is an integer.

```
AggregateExpectedAccidentsPlot[modelsev_List,
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = Min[(dateconv[#1⟦4⟧] &) /@modelsev] - 1;
  plotMax = Max[(dateconv[Last[#1]] &) /@modelsev] + 1;
  Plot[Evaluate[Plus @@ (expAcc[#1, plotMin, days] &) /@modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Cumulative Accidents Expected", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of `AggregateExpectedAccidentsPlot` below plots the aggregate expected number of accidents of all the SSRAs in *modelsev* from a specified start date to a specified ending date. This version will apply when the first argument supplied is a list, the second is a list containing two vectors of length three and the third is an integer.

```
AggregateExpectedAccidentsPlot[modelsev_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
  yearInc_Integer, opts___?OptionQ] := Module[{plotMin, plotMax},
  plotMin = dateconv[startDate] - 1; plotMax = dateconv[endDate] + 1;
  Plot[Evaluate[Plus @@ (expAcc[#1, plotMin, days] &) /@modelsev],
   {days, plotMin, plotMax}, opts, FrameLabel → {"Calendar Year",
     "Cumulative Accidents Expected", "SSRA Aggregate", None},
   FrameTicks → {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

opts is an optional argument for specifying `Plot` options.

ExpectedAccidentsTrendPlot[]

The version of `ExpectedAccidentsTrendPlot` below plots the n-year aggregate expected-accidents trend from the start of the earliest SSRA until a specified date (e.g., today).

```
ExpectedAccidentsTrendPlot[modelsev_List,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3, year_Integer,
   yearInc_Integer, (opts___)?OptionQ] := Module[{plotMin, plotMax},
   plotMin = Min[(dateconv[#1[[4]]] & ) /@ modelsev];
    plotMax = dateconv[endDate];
    Plot[Evaluate[Plus @@ (expAcc[#1, days, days + (year*1461)/4] & )
/@
        modelsev], {days, plotMin, plotMax}, opts,
     FrameLabel -> {"Calendar Year", ToString[StringForm[
         "Accidents Expected in Next `1` Year(s)", year]], "SSRA
Aggregate", None},
     FrameTicks -> {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

The version of ExpectedAccidentsTrendPlot below plots the n-year expected-accident trend from a specified start date to a specified end date.

```
ExpectedAccidentsTrendPlot[modelsev_List,
   {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
    endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
year_Integer,
   yearInc_Integer, (opts___)?OptionQ] := Module[{plotMin, plotMax},
   plotMin = dateconv[startDate]; plotMax = dateconv[endDate];
    Plot[Evaluate[Plus @@ (expAcc[#1, days, days + (year*1461)/4] & )
/@
        modelsev], {days, plotMin, plotMax}, opts,
     FrameLabel -> {"Calendar Year", ToString[StringForm[
         "Accidents Expected in Next `1` Year(s)", year]], "SSRA
Aggregate", None},
     FrameTicks -> {yearticks[plotMin, plotMax, yearInc], Automatic}]]
```

opts is an optional argument for specifying Plot options.

FlightHoursTest[]

This function tests SSRAs to ensure it is structured correctly. *test1* ensures that the length is an odd integer greater than or equal to 3. *test2* determines whether the even elements 2 and up are non-negative numbers. This doesn't ensure that these numbers aren't Complex. *test3* determines whether odd elements 1 and up are vectors of length 3. *test4* determines whether the first sub-element of odd elements 1 and up are integers between 1900 and 2100. *test5* determines whether the second sub-element of odd elements 1 and up are integers between 1 and 12. *test6* determines whether the third sub-element of odd elements 1 and up are integers between 1 and 31. This still doesn't completely guarantee a real date since it is possible that one might input Feb 29 (non leap year), 30 or 31, Apr 31, Jun 31, Sep 31 or Nov 31. *test7* determines whether odd elements 1 and up are chronologically-ordered but doesn't flag situations where two adjacent dates are equal. *test8* determines whether two adjacent dates are equal.

```
FlightHoursTest[FHdata_List] :=
 Module[{len, test1, test2, test3, test4, test5, test6, test7},
  len = Length[FHdata]; test1 = OddQ[len] && len ≥ 3;
  test2 = And @@ (NumberQ[FHdata[[#1]]] && NonNegative[FHdata[[#1]]] &) /@
     Range[2, len, 2]; test3 = And @@
    (VectorQ[FHdata[[#1]]] && Length[FHdata[[#1]]] == 3 &) /@ Range[1, len, 2];
  test4 = And @@ (IntegerQ[FHdata[[#1, 1]]] && 1900 < FHdata[[#1, 1]] < 2100 &) /@
     Range[1, len, 2]; test5 = And @@
    (IntegerQ[FHdata[[#1, 2]]] && 1 ≤ FHdata[[#1, 2]] ≤ 12 &) /@ Range[1, len, 2];
  test6 = And @@ (IntegerQ[FHdata[[#1, 3]]] && 1 ≤ FHdata[[#1, 3]] ≤ 31 &) /@
     Range[1, len, 2]; test7 =
   And @@ (OrderedQ[FHdata[[#1]] < FHdata[[#1 + 2]]] &) /@ Range[1, len - 2, 2];
  test8 = And @@ (FHdata[[#1]] =!= FHdata[[#1 + 2]] &) /@ Range[1, len - 2, 2];
  Which[! test1, Message[FlightHoursTest::"fhourslength"],
   ! test2, Message[FlightHoursTest::"fhoursnonnegnumbers"],
   ! test3, Message[FlightHoursTest::"fhoursdatelength"],
   ! test4, Message[FlightHoursTest::"fhoursdateyear"],
   ! test5, Message[FlightHoursTest::"fhoursdatemonth"],
   ! test6, Message[FlightHoursTest::"fhoursdateday"], ! test7,
   Message[FlightHoursTest::"fhoursdatechrono"], ! test8,
   Message[FlightHoursTest::"fhoursdateequal"], 1 == 1, True]]
```

FlightHours[]

FlightHours calculates the quantity of flight hours between *startDate* and *endDate* given the flight-hour data in *FHdata*. *FHdata* must be structured as a list containing an odd number of elements greater than or equal to three. The odd-numbered elements must be dates in {y, m, d} format. The even-numbered elements must be quantities of flight hours to be accrued between the preceding and succeeding dates.

```
FlightHours[FHdata_List,
  {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
   endDate_ /; VectorQ[endDate] && Length[endDate] == 3}] :=
 fHours[FHdata, dateconv[startDate], dateconv[endDate]]
```

FlightHoursDate[]

FlightHoursDate calculates, starting from *startDate*, the date when the specified quantity of flight hours (*fhrs*) will be accumulated. This function relies on the auxiliary function FHdate (except for date conversions). Refer to the description of that function for further information on how it works.

```
FlightHoursDate[FHdata_List, startDate_ /;
   VectorQ[startDate] && Length[startDate] == 3, fhrs_Integer] :=
 Take[ToDate[24 60 60 FHdate[FHdata, dateconv[startDate], fhrs]], 3]
```

ExpectedAccidentsNext[]

ExpectedAccidentsNext calculates, starting from *startDate*, the aggregate number of accidents expected to occur in the next *fhrs* flight hours. This function relies on the auxiliary function expaccnext (except for date conversions). Refer to the description of that function for further information on how it works.

```
ExpectedAccidentsNext[modelsev_List, FHdata_List,
  startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
  fhrs_Integer] := expaccnext[modelsev, FHdata, dateconv[startDate], fhrs]
```

ExpectedAccidentsNextPlot[]

ExpectedAccidentsNextPlot plots, between *startDate* and *endDate*, the aggregate number of accidents expected in the next *fhrs* flight hours. This function relies on the auxiliary function expaccnext. Refer to the description of that function for further information on how it works. *yearInc* specifies the calendar-year increments to be used on the plot. *opts* is an optional argument for specifying Plot options.

```
ExpectedAccidentsNextPlot[modelsev_List, FHdata_List,
   {startDate_ /; VectorQ[startDate] && Length[startDate] == 3,
    endDate_ /; VectorQ[endDate] && Length[endDate] == 3},
fhrs_Integer,
   yearInc_Integer, (opts___)?OptionQ] := Module[{plotMin, plotMax},
   plotMin = dateconv[startDate]; plotMax = dateconv[endDate];
    Plot[expaccnext[modelsev, FHdata, days, fhrs], {days, plotMin,
plotMax},
     opts, PlotPoints -> 10, MaxBend -> 1000, PlotRange -> {0,
Automatic},
     FrameLabel -> {"CalendarYear", ToString[StringForm[
         "Accidents in Next `1` Flight Hours", fhrs]], "SSRA
Aggregate",
       None}, FrameTicks -> {yearticks[plotMin, plotMax, yearInc],
       Automatic}]]
```

- **Definitions for system functions**

None.

- **Restore protection of system symbols**

Not applicable.

- **End the private context**

```
End[]
```

Epilog

This section protects exported symbols and ends the package.

- **Protect exported symbol**

```
Protect[SSRATest, SSRASelect, FirstSSRA, LastSSRA, OccurrenceRate,
OccurrenceRateTable, OccurrenceRatePie, OccurrenceRatePlot,
AggregateOccurrenceRatePlot, ExpectedAccidents, ExpectedAccidentsTable,
ExpectedAccidentsPie, ExpectedAccidentsPlot,
AggregateExpectedAccidents, AggregateExpectedAccidentsPlot,
ExpectedAccidentsTrendPlot, FlightHoursTest, FlightHours,
FlightHoursDate, ExpectedAccidentsNext, ExpectedAccidentsNextPlot]
```

- **End the package context**

```
EndPackage[]
```

Appendix B

Usage of Expected Accident Aggregation Model Functions

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix B

Usage of Expected Accident Aggregation Model Functions

This appendix contains usage messages for each of the new add-on functions in the package for the *ExpectedAccidentAggregationModel*. These usage messages are an important utility since an analyst can access them whenever a reminder is needed concerning what each function does and what its syntax requirements are. The application of many of these functions may be found in Chapters 3 and 4.

First the package is loaded:

```
Needs["AviationSafety`ExpectedAccidentAggregationModel`"]
```

The usage message for this package is:

```
?ExpectedAccidentAggregationModel

ExpectedAccidentAggregationModel.m (version 1.0.1) is a package that
  contains functions for the Expected Accident Aggregation model.
```

The new add-on functions, in alphabetical order, are:

```
usagelist = ?AviationSafety`ExpectedAccidentAggregationModel`*

AggregateExpectedAccidents       FirstSSRA
AggregateExpectedAccidentsPlot   FlightHours
AggregateOccurrenceRatePlot      FlightHoursDate
ExpectedAccidentAggregationModel FlightHoursTest
ExpectedAccidents                LastSSRA
ExpectedAccidentsNext            OccurrenceRate
ExpectedAccidentsNextPlot        OccurrenceRatePie
ExpectedAccidentsPie             OccurrenceRatePlot
ExpectedAccidentsPlot            OccurrenceRateTable
ExpectedAccidentsTable           SSRASelect
ExpectedAccidentsTrendPlot       SSRATest
```

Their usages messages are:

? AggregateExpectedAccidents

AggregateExpectedAccidents[SSRAlist, {startDate, endDate}] aggregates all SSRAs in SSRAlist from startDate to endDate. AggregateExpectedAccidents[SSRAlist, nameList, {startDate, endDate}] aggregates only SSRAs in nameList from startDate to endDate. AggregateExpectedAccidents[SSRAlist, startDate, years] aggregates all SSRAs in SSRAlist from startDate forward the specified number of years. AggregateExpectedAccidents[SSRAlist, nameList, startDate, years] aggregates only SSRAs in nameList from startDate forward.

? AggregateExpectedAccidentsPlot

AggregateExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots the aggregate of all SSRAs in SSRAlist from a day before the earliest begins to a day after the last ends. yearInc specifies the calendar-year increments to be used on the plot. AggregateExpectedAccidentsPlot[SSRAlist, {startDate, endDate}, yearInc, opts] plots the aggregate of all SSRAs from startDate to endDate. AggregateExpectedAccidentsPlot[SSRAlist, nameList, yearInc, opts] plots the aggregate of only SSRAs in nameList from a day before the earliest begins to a day after the last ends. AggregateExpectedAccidentsPlot[SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots the aggregate of only SSRAs in nameList from startDate to endDate. opts is an optional argument for specifying Plot options.

? AggregateOccurrenceRatePlot

AggregateOccurrenceRatePlot[SSRAlist, yearInc, opts] plots the aggregated occurrence rate (accidents per day) of all SSRAs in SSRAlist from a day before the earliest begins to a day after the last ends. yearInc specifies the calendar-year increments to be used on the plot. AggregateOccurrenceRatePlot[SSRAlist, {startDate, endDate}, yearInc, opts] plots the aggregated occurrence rate of all SSRAs from startDate to endDate. AggregateOccurrenceRatePlot[SSRAlist, nameList, yearInc, opts] plots the aggregated occurrence rate of only SSRAs in nameList from a day before the earliest begins to a day after the last ends. AggregateOccurrenceRatePlot[SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots the aggregated occurrence rate of only SSRAs in nameList from startDate to endDate. opts is an optional argument for specifying Plot options.

?ExpectedAccidents

```
ExpectedAccidents[SSRAlist, name, {startDate,
  endDate}] calculates the expected number of accidents
  from startDate to endDate of the named SSRA from SSRAlist.
```

?ExpectedAccidentsNext

```
ExpectedAccidentsNext[SSRAlist, FHdata, startDate, fhrs] calculates,
  starting from startDate, the aggregate number of accidents
  expected to occur in the next fhrs flight hours, given the
  SSRA data in SSRAlist and the flight-hours data in FHdata.
```

?ExpectedAccidentsNextPlot

```
ExpectedAccidentsNextPlot[SSRAlist, FHdata, {startDate,
  endDate}, fhrs, yearInc, opts] plots, between startDate
  and endDate, the aggregate number of accidents expected to
  occur in the next fhrs flight hours, given the SSRA data
  in SSRAlist and the flight-hours data in FHdata. yearInc
  specifies the calendar-year increments to be used on the plot.
  opts is an optional argument for specifying Plot options.
```

?ExpectedAccidentsPie

```
ExpectedAccidentsPie[SSRAlist, {startDate, endDate}, topSSRAs, opts]
  generates a pie chart of the expected number of accidents
  from startDate to endDate for the SSRAs in SSRAlist. Only
  the largest SSRAs, as specified by topSSRAs, are given their
  own wedges. The remaining SSRAs are placed in a single wedge.
  opts is an optional argument for specifying PieChart options.
```

?ExpectedAccidentsPlot

ExpectedAccidentsPlot[SSRAlist, yearInc, opts] plots the expected number of accidents for all SSRAs in SSRAlist from a day before the earliest begins to a day after the last ends. yearInc specifies the calendar-year increments to be used on the plot. ExpectedAccidentsPlot[SSRAlist, {startDate, endDate}, yearInc, opts] plots the expected number of accidents for all SSRAs from startDate to endDate. ExpectedAccidentsPlot[SSRAlist, name, yearInc, opts] plots the expected number of accidents for the named SSRA from a day before it begins to a day after it ends. ExpectedAccidentsPlot[SSRAlist, name, {startDate, endDate}, yearInc, opts] plots the expected number of accidents for the named SSRA from startDate to endDate. ExpectedAccidentsPlot[SSRAlist, nameList, yearInc, opts] plots the expected number of accidents for only SSRAs in nameList from a day before the earliest begins to a day after the last ends. ExpectedAccidentsPlot[SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots the expected number of accidents for only SSRAs in nameList from startDate to endDate. opts is an optional argument for specifying Plot options.

?ExpectedAccidentsTable

ExpectedAccidentsTable[SSRAlist, startDate, year, opts] generates a table of n-year expected accidents for all SSRAs in SSRAlist from startDate forward. ExpectedAccidentsTable[SSRAlist, nameList, startDate, year, opts] generates a table of n-year expected accidents for only SSRAs in nameList from startDate forward. ExpectedAccidentsTable[SSRAlist, startDate, yearList, opts] generates a table for all SSRAs with multiple expected-accident columns where the years are specified by yearList. ExpectedAccidentsTable[SSRAlist, nameList, startDate, yearList, opts] generates a table with multiple expected-accident columns for only SSRAs in nameList. opts is an optional argument for specifying TableForm options.

?ExpectedAccidentsTrendPlot

ExpectedAccidentsTrendPlot[SSRAlist, endDate, year, yearInc, opts] plots the n-year aggregate expected-accident trend for all SSRAs in SSRAlist from the beginning of the earliest SSRA until endDate. yearInc specifies the calendar-year increments to be used on the plot. ExpectedAccidentsTrendPlot[SSRAlist, {startDate, endDate}, year, yearInc, opts] plots the n-year expected accident trend for all SSRAs from startDate to endDate. opts is an optional argument for specifying Plot options.

?FirstSSRA

```
FirstSSRA[SSRAlist] selects from SSRAlist the SSRA that began first.
```

?FlightHours

```
FlightHours[FHdata, {startDate, endDate}]
  calculates the number of flight hours accrued between
  startDate to endDate given the flight-hours data in FHdata.
```

?FlightHoursDate

```
FlightHoursDate[FHdata, startDate, fhrs}] calculates,
  starting from startDate, the date when fhrs flight hours
  will be accumulated, given the flight-hours data in FHdata.
```

?FlightHoursTest

```
FlightHoursTest[FHdata] tests a flight-
  hours list to ensure it's structured correctly.
```

?LastSSRA

```
LastSSRA[SSRAlist] selects from SSRAlist the SSRA that began last.
```

?OccurrenceRate

```
OccurrenceRate[SSRAlist, name, date]
  calculates the occurrence rate (accidents per day) on
  the specified date for the named SSRA from SSRAlist.
```

?OccurrenceRatePie

```
OccurrenceRatePie[SSRAlist, date, topSSRAs, opts] generates
  a pie chart of occurrence rates in accidents per day on the
  specified date for the SSRAs in SSRAlist. Only the largest
  SSRAs, as specified by topSSRAs, are given their own wedges.
  The remaining SSRAs are placed in a single wedge.  opts
  is an optional argument for specifying PieChart options.
```

? OccurrenceRatePlot

OccurrenceRatePlot[SSRAlist, yearInc, opts] plots the occurrence rates (accidents per day) of all SSRAs in SSRAlist from a day before the earliest begins to a day after the last ends. yearInc specifies the calendar-year increments to be used on the plot. OccurrenceRatePlot[SSRAlist, {startDate, endDate}, yearInc, opts] plots the occurrence rates of all SSRAs from startDate to endDate. OccurrenceRatePlot[SSRAlist, name, yearInc, opts] plots the occurrence rate of the named SSRA from a day before it begins to a day after it ends. OccurrenceRatePlot[SSRAlist, name, {startDate, endDate}, yearInc, opts] plots the occurrence rate of the named SSRA from startDate to endDate. OccurrenceRatePlot[SSRAlist, nameList, yearInc, opts] plots the occurrence rates of only SSRAs in nameList from a day before the earliest begins to a day after the last ends. OccurrenceRatePlot[SSRAlist, nameList, {startDate, endDate}, yearInc, opts] plots the occurrence rates of only SSRAs in nameList from startDate to endDate. opts is an optional argument for specifying Plot options.

? OccurrenceRateTable

OccurrenceRateTable[SSRAlist, date, opts] generates a table of occurrence rates in accidents per day on the specified date for all SSRAs in SSRAlist. OccurrenceRateTable[SSRAlist, nameList, date, opts] generates a table of occurrence rates for only SSRAs in nameList. opts is an optional argument for specifying TableForm options.

? SSRASelect

SSRASelect[SSRAlistAll, sevlev] selects all SSRAs in SSRAlistAll with severity level sevlev.

? SSRATest

SSRATest[ssra] tests an SSRA to ensure it's structured correctly.

Appendix C

Distribution List

THIS PAGE INTENTIONALLY LEFT BLANK

Appendix C

Distribution List

No. of Copies	Organization
7	Defense Technical Information Center ATTN: DTIC-OMI 8725 John J. Kingman Road, Suite 0944 Fort Belvoir, VA 22060-6218
15	Director US Army Materiel Systems Analysis Activity ATTN: AMXSY- A (M. Cushing, D. Mortin) AMXSY-DDS (3 cys) 392 Hopkins Road Aberdeen Proving Ground, MD 21009-5071

THIS PAGE INTENTIONALLY LEFT BLANK.